ULTRIX-32 Guide to the BIND Service

Order No. AA-LY21A-TE

ULTRIX-32 Operating System, Version 3.0

Digital Equipment Corporation

This manual was written and produced by the ULTRIX Documentation Group in Nashua, New Hampshire.

Contents

About This Guide

1 Introduction to the BIND Service

2 Setting Up the BIND Service

B Appendix

C Appendix

D Appendix

Figures

About This Guide

This guide provides introductory information about the the Berkeley Internet Name Domain (BIND) service and explains how to install and troubleshoot the service. In addition, this guide will assist you in developing BIND management procedures by presenting guidelines from which you can develop specific procedures for your site.

Audience

This guide is meant for the person responsible for maintaining networks and system utilities such as mail on the ULTRIX operating system. This person is usually the system manager, but could be a network manager or the system manager who is also a user of a MicroVAX processor running the ULTRIX operating system. This guide assumes that the reader is familiar with the ULTRIX system commands, the system configuration, the naming conventions, and an editor such as vi or ed. It also assumes that the reader knows the names and addresses of the other systems on the local network.

Organization

This guide consists of four chapters, several appendixes, and an index. The chapters are:

Chapter 1: Introduction

This chapter introduces the BIND service. It provides the background information that you need before you can set up and run BIND on your system. This chapter also describes basic BIND concepts.

Chapter 2: Setting Up the BIND Service

This chapter describes how to perform the preliminary BIND setup on your system using the bindsetup command, how to set up the BIND service manually, and how to edit the BIND-related files.

Because the BIND environment varies from site to site, you need to edit these files regardless of whether you use the bindsetup command. The description of how to set up the BIND service manually is included for those who want to understand how the BIND service

operates in addition to understanding how to edit the BIND-related files.

Chapter 3: Managing and Using the BIND Service
This chapter describes how to manage the BIND service, including defintions of the BIND administrative roles and information on how to register a new top-level domain. It also describes how to use some of the BIND features, such as obtaining a host name and IP address using the nslookup command.

Chapter 4: Troubleshooting the BIND Service
This chapter describes how to debug the BIND service and review the resulting debug files. It also offers general suggestions on how to troubleshoot the BIND service.

Appendix A
This appendix provides sample BIND files for your reference. Some depict a system with the BIND service set up properly, while others indicate error conditions with the service.

Appendix B
This appendix shows a sample ftp session for transferring the BIND registration questionnaire from the Network Information Center (NIC). It also contains a copy of the registration questionnaire for your information.

Appendix C
This appendix lists several papers, articles, and RFCs related to the BIND service which you may want to read.

Appendix D
This appendix provides a sample interactive session with the nslookup command. The sample is intended to help you get started with using the command.

Related Documents

You should have available the related hardware documentation for your system. You also should have the other documents in the ULTRIX documentation set.

Conventions

The following conventions are used in this guide:

special	In text, each mention of a specific command, option, partition, pathname, directory, or file is presented in this type.

command(x)	In text, cross-references to the command documentation include the section number in the reference manual where the commands are documented. For example: See the cat(1) command. This indicates that you can find the material on the cat command in Section 1 of the ULTRIX Reference Pages.
literal	In syntax descriptions, this type indicates terms that are constant and must be typed just as they are presented.
italics	In syntax descriptions, this type indicates terms that are variable.
[ ]	In syntax descriptions, square brackets indicate terms that are optional.
. . .	In syntax descriptions, a horizontal ellipsis indicates that the preceding item can be repeated one or more times.
function	In function definitions, the function itself is shown in this type. The function arguments are shown in italics.
UPPERCASE	The ULTRIX system differentiates between lowercase and uppercase characters. Enter uppercase characters only where specifically indicated by an example or a syntax line.
example	In examples, computer output text is printed in this type.
example	In examples, user input is printed in this bold type.
%	This is the default user prompt in multiuser mode.
#	This is the default superuser prompt.
>>>	This is the console subsystem prompt.
.	In examples, a vertical ellipsis indicates that not all of the
.	lines of the example are shown.
.	
KEYNAME	In examples, a word or abbreviation in angle brackets indicates that you must press the named key on the terminal keyboard.
CTRL/x	In examples, symbols like this indicate that you must hold down the CTRL key while you type the key that follows the slash. Use of this combination of keys may appear on your terminal screen as the letter preceded by the circumflex character. In some instances, it may not appear

Introduction to the BIND Service 1

This chapter provides an overview of the Berkeley Internet Name Domain (BIND) Service.

The Bind service allows client systems to obtain host names and addresses from BIND servers, and is basically a host name and address lookup service for information on the Internet network. You can use the BIND service to replace or supplement the host table mapping provided by the local /etc/hosts file or the Yellow Pages (YP) service.

The BIND service is comprised of two parts, the software interface and the server. The software interface is called the resolver, which consists of a group of routines that reside in the C library /usr/lib/libc.a. The resolver exchanges query packets with a BIND server. All BIND servers have a name server daemon running in the background, which services queries on a given network port. The standard port for UDP and TCP is specified in the /etc/services file.

An advantage of using the BIND service instead of the host table lookup method for host name and address resolution is that you avoid the need for a single centralized clearing house for all the names and addresses. With the BIND service you can delegate the authority to disseminate host information to the different systems on the network responsible for it. This works well for large networks where systems cross organizational boundaries.

The BIND service utilizes several concepts such as domains, zones, servers, clients, and host names and addresses. The rest of this chapter introduces these concepts and summarizes the steps the BIND service takes to resolve a query. For a complete discussion of host names and Internet addresses, see the Guide to Networking.

The BIND service breaks the Internet into a hierarchy of domains, similar to a tree structure. Each domain is given a label, and The name of the domain is the concatenation of all the labels of the domains, from the root to the current domain, listed from right to left and separated by dots. A label must be unique within its domain. The entire BIND Internet hierarchy is partitioned into several zones, each starting at a domain and extending down to the leaf domains, or to domains where other zones start. A zone is a subdivision of a domain and is a discrete, non-

overlapping entity. Each zone is an area of authority for which a master server is responsible, and therefore usually represents an administrative boundary.

Currently there are seven top-level domains in the BIND hierarchy in the United States:

arpa	For the Arpanet (gradually being phased out)
com	For commercial institutions
edu	For educational institutions
gov	For the government
mil	For military organizations
net	For network-type organizations such as network service centers, consortia, and information centers
org	For miscellaneous organizations such as professional societies, similar non-profit organizations, and so forth

In addition to these, there are several top-level domains for individual countries. You can contact the NIC for more information about them.

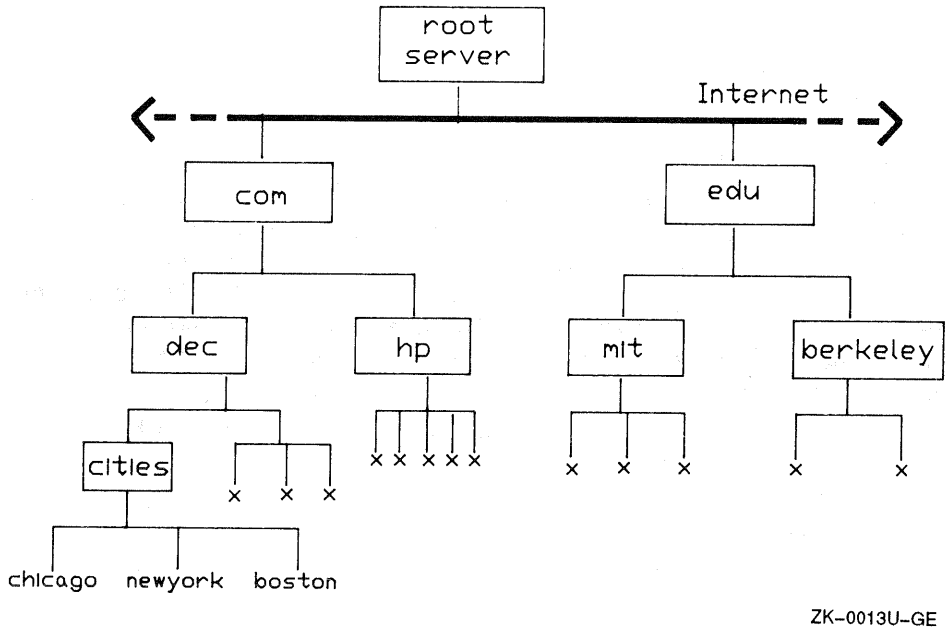

Figure 1-1: Hierarchy of BIND Zones and Domains on the Internet

Figure 1-1 shows the hierarchy of the Internet, two top-level domains, and some of the major zones. In Figure 1-1, everything below com is in the com top-level domain, and the zones are mit.edu, dec.com, and cities.dec.com, and the host names (sometimes termed leaf domains) have the names of cities or are depicted by an x.

Assuming a host name in the zone cities.dec.com is chicago, the following is the fully qualified domain name for that host:

```
chicago.cities.dec.com.
```

In this example, com is the top level domain, cities.dec.com a subdomain of com, and chicago is a host name. If a master server has the authority for the dec.com domain only, then dec.com is a zone.

In the preceding example, note the dot (.) at the end of the domain name. This indicates that the domain name is fully qualified, and is thus a complete, definitive, and absolute name of a singular host.

The rest of this chapter introduces BIND servers and clients and indicates how an individual can make use of the BIND service.

1.1 BIND Servers

A BIND server is a system running the named daemon and therefore can answer BIND queries. There are several types of BIND servers: root, master, caching, forwarding, and slave. The following sections describe each them in detail.

1.1.1 The Root Server

Root servers are the ultimate authorities. The root servers know about all the top-level domains on the Internet network. From these top-level domains, information can be gathered about hosts on subdomains. The root servers, for example, do not necessarily know about the cities.dec.com subdomain. However, by performing an NS query with the nslookup command, a root server can tell you to check with decwrl.dec.com for information about a host on the cities.dec.com subdomain.

If a client requests information on another domain, any server, other than slave, can pass along the request to a root server.

At this time there are seven root servers in the continental United States. These root servers are:

ns.nasa.gov.
sri-nic.arpa.
a.isi.edu.
gunter-adam.arpa.
brl-aos.arpa.
terp.umd.edu.
c.nyser.net.

The period (.) at the end of each root server name indicates that this is the absolute pathname and that no BIND name extensions are to be appended. Without the period, the server name is relative to the current domain.

With the proper option set in a BIND server's cache file, the BIND service automatically updates the server's cache with information about any changes regarding the root servers. This information will be absorbed by the boot file when your server reboots. This is described in Chapter 2.

The Network Information Center (NIC) determines who will be root servers. The toll-free number for the NIC is:

(800) - 235 - 3155

The electronic mailing address is:

hostmaster@sri-nic.arpa

1.1.2 The Master Server

A master server is the authority for the current domain space and maintains the BIND data bases for its zone. A server may be a master server for multiple domains, being the primary server for some domains and a secondary server for others.

The primary master server loads its data base from a file on disk. This server can also delegate to other servers in its zone the authority to answer queries for its domain space.

A secondary master server receives its authority and its data base from the primary master server. When a secondary master server first boots, it loads the data for the zone from a backup file, if possible (assuming you configured your BIND service this way). It then consults with a primary master server to check that the data base is still up to date. After the secondary master server is running, it periodically checks with the primary master server to see if it needs to update its data base. For information on how to define the frequency of the update checks, see Chapter 2.

Each BIND domain should have at least two master servers, one primary and one or more secondary. The duplicate secondary servers act as backup servers in the event that the primary master server fails, is overloaded, or is down.

1.1.3 The Caching Server

All servers cache the information they receive for use until the data expires. However, caching servers have no authority for any zone, and thus have no data bases to maintain. These servers service BIND queries by asking other servers who have authority, such as a master server, for the information. Caching servers store the information in a cache until the data expires. The expiration date is based on a time to live (ttl) field, which is attached to the data when the caching server receives it.

1.1.4 The Forwarding Server

Forwarding servers, called forwarders, process recursive requests that slave servers cannot resolve locally. A forwarder can be any BIND server that has Internet access. Thus, forwarders can be a primary or a secondary master server or a caching only server. The configuration files on the slave servers define which systems the slaves will access as forwarders.

Forwarders have full access to the Internet network and therefore are able to obtain information not held in their local caches from root servers.

Because forwarders receive many requests from slave servers, they tend to have a larger local cache than do slave servers. All the hosts on the domain benefit from this meta-cache, which reduces the total number of queries from that site by forwarding them to the root servers on the outside Internet network.

A slave server and forwarder configuration is typically used when you do not want all the servers to interact with the rest of the servers on the Internet network. For example, assume a site consists of several workstations and a VAX 8800 processor acting as a BIND forwarder. Assume the workstations are not to have access to the Internet network. To give the workstations the appearance of access to the Internet network, they could be set up as BIND slave servers to the VAX 8800 system. In this case, the BIND forwarder forwards the workstations' queries and interacts with other BIND servers on the Internet network. When the forwarder resolves the queries, it sends the answer to the slave server. Figure 1-2 shows the relationship among master and slave servers and forwarders. The arrows in Figure 1-2 depict the general flow of information to and from some of the hosts.

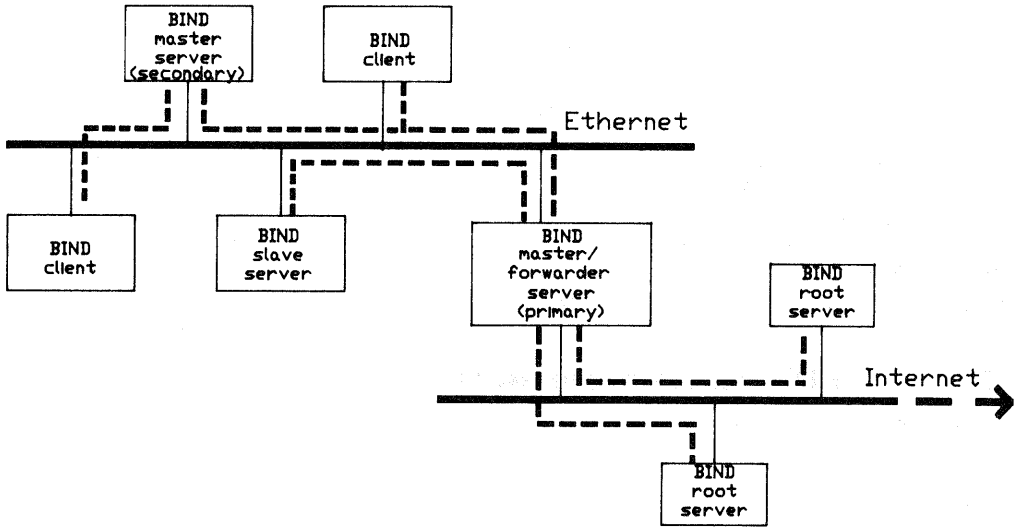

Figure 1-2: Relationship of Master/Forwarder and Slave Servers

Note

You can run the BIND service on a local network, only, without having a forwarder on your network. However, if you do not have a forwarder on your network there is no need to have slave servers. Without forwarders, your system does not have access to the root servers on the Internet.

1.1.5 The Slave Server

Slave servers typically do not have full network access and therefore cannot directly interact with root servers if the information requested is not in their local caches.

If a slave server cannot resolve a query locally, it forwards the query to its fixed list of forwarders. The slave servers try the forwarders listed in their boot files, one at a time, until the list is exhausted or the query is

satisfied.

1.2 BIND Clients

A BIND client is any system that uses the BIND service to resolve host names and addresses. BIND clients make queries, but they never resolve them locally. Instead, BIND servers resolve the client's requests.

BIND clients do not run the named daemon. Instead, BIND clients have the resolver file /etc/resolv.conf. No other BIND files are necessary. Here is an example of a /etc/resolv.conf file:

```
domain dec.com
nameserver 128.11.22.33
nameserver 128.11.22.44
```

The /etc/resolv.conf file tells the resolver the IP address of the BIND servers which can service the client's BIND requests.

1.3 How BIND Queries are Resolved

The following steps describe the usual procedure a slave server and forwarder take to resolve a BIND query:

1. A slave server receives a query for a host name resolution.
2. The slave server uses the gethostbyname library routine.
3. If the gethostbyname library routine cannot obtain the information, the slave server asks the forwarders listed in its BIND boot file (the default is named.boot) one at a time, until the query is resolved or the list is exhausted.
4. If the forwarder does not have the information in its local cache, it asks the root servers listed in its BIND data file, one at a time, until the query is resolved or the list is exhausted.
5. The root server provides the forwarder with the information needed to contact servers of the domain space containing the host in question.
6. The forwarder sends a request to a server for that domain. It gets the server's address information from a root server.
7. The server provides the forwarder with the information to contact servers of the next lower domain.
8. Steps 5 and 6 repeat until the forwarder actually gets the host information, or until the information obtained from the root server is exhausted.
9. The forwarder returns the results to the slave server, even if the

Other BIND servers follow similar steps.

1.4 How to Use the BIND Service

If the BIND service is enabled, your system automatically uses it for any process that needs a host name or address such as mail, rlogin, ftp, and so forth.

In addition, users on a system with the BIND service properly set up can run the nslookup and nsquery commands to find host names and addresses. See Chapter 3 and nslookup(1) and nsquery(1) in the ULTRIX Reference Pages for further information.

Setting Up the BIND Service 2

This chapter explains how to perform the preliminary setup of the Internet Name Domain (BIND) service using the bindsetup command, and how to perform the preliminary setup manually. The bindsetup command allows you to set up your system as a BIND server or client. See bindsetup(8) in the ULTRIX Reference Pages for further information about the bindsetup command. See Chapter 1 for a description of BIND servers and clients.

Note

After you have installed your ULTRIX operating system and set up the BIND service, you need to edit the sendmail configuration file /etc/sendmail.cf. For the mail utility to run with the BIND service, you must specify your BIND domain in the sendmail.cf file. For example, if your domain name is cities.dec.com, here is the appropriate entry in the sendmail.cf file:

```
DDcities.dec.com
```

See the sendmail documentation and read the comments in the file itself for further information.

The topics discussed in this chapter are:

- Setting up a BIND client automatically
- Setting up a BIND client manually
- Setting up a BIND server automatically
- Setting up a BIND server manually
- Format of BIND file entries

If you want the BIND service to be able to resolve queries about other domains, you must register your domain. Chapter 3 describes how to register your domain with a public network.

2.1 Setting Up a BIND Client Automatically

To set up your system as a BIND client automatically, run the bindsetup command and then reboot the system. However, before you run the bindsetup command, be sure that the system is in multiuser mode and that the network is up. The easiest way to run the bindsetup command is to supply the domain name and server IP address on the command line. For example, if the domain name is cities.dec.com and the server IP addresses are 128.11.22.33 and 128.11.22.44 type:

```
# bindsetup -c cities.dec.com 128.11.22.33 128.11.22.44
```

The bindsetup command then sets up your system silently as a BIND client with the domain and servers specified.

To use the bindsetup command interactively to set up a BIND client, type:

```
# bindsetup
```

The bindsetup command then displays a menu and prompts you for the required information.

After the setup has been completed, the bindsetup command lists the updated files for your information. You should then reboot the system. This ensures that the BIND service has your system's fully qualified domain name, such as chicago.cities.dec.com. The following command reboots your system:

```
# /etc/shutdown -r now
```

Note

If the bindsetup command fails for any reason, be sure to check the /etc/svcorder file. If the BIND service is not properly set up, be sure that there is no bind entry in this file.

When prompted for the domain, be sure to supply the domain name without the trailing dot (.), for example:

cities.dec.com

2.2 Setting Up A BIND Client Manually

To set up your system as a BIND client manually, you need to follow these three steps:

1. Create the file /etc/resolv.conf
2. Set the host name in the /etc/hosts and /etc/rc.local files

3. Edit the /etc/svcorder file
4. Reboot the system

2.2.1 Create the Resolver File

The resolver file /etc/resolv.conf designates the BIND servers on the network that can answer queries.

Note

If your system is a BIND server running the named daemon, you should not set up your system as a client, since the /etc/resolv.conf file, if it exists, is read each time the gethostbyname() or gethostbyaddr() routine is called.

An easy way to tell if your system is a BIND server is to see if the named daemon is running:

```
# ps -aux | grep named
```

The resolv.conf file consists of at least two entries. The first entry defines the domain and the second entry defines the server. It is best to have additional entries, one for each additional server. Server replication reduces the possibility of the BIND service being interrupted in the event that a server goes down. Here is the format for /etc/resolv.conf file entries:

domain	*domainname*
nameserver	*IP address*

For example, the following shows the contents of the resolver file for a client on the domain cities.dec.com. In this example there are two servers listed. Note that the semicolon (;) designates a comment line in BIND files:

```
;
;
;
domain          cities.dec.com
nameserver      128.11.22.33
nameserver      128.11.22.44
```

See resolver(5) in the ULTRIX Reference Pages for further information about the /etc/resolv.conf file.

2.2.2 Set the Host Name

For information on how to set the host name in the /etc/hosts and /etc/rc.local files, see Section 2.4.3.

2.2.3 Edit the Services Order File

For information on how to edit the services order file /etc/svcorder, see Section 2.4.4.

2.2.4 Reboot the System

After you have created the resolver file, modified your host name to the fully qualified BIND name, and edited the services order file, you should reboot the system. By rebooting the system, you cause all the modifications to the files to take effect.

The following command shuts down the system and reboots it immediately:

```
# /etc/shutdown -r now
```

See shutdown(8) in the ULTRIX Reference Pages for further information about shutting down the system.

2.3 Setting Up a BIND Server Automatically

To set up your system as a BIND server automatically, you need to edit a few files, and run the bindsetup command, and reboot the system. However, before you run the bindsetup command, be sure that the system is in multiuser mode and that the network is up. These are the steps for setting up a BIND server:

1. Run the bindsetup command and answer the questions
2. Edit the domain data files
3. Reboot the System

2.3.1 Run the bindsetup Command

The first step is to execute the bindsetup command, specifying no options. First, however, be sure that the network is up.

```
# bindsetup
```

The bindsetup command prompts you for the following information:

- The domain name.
- The type of configuration, such as primary master, secondary master, caching, or slave server.

- The full path name of the directory where the BIND data files are to reside: the default is /etc/namedb.
- The BIND boot file name: the default is /etc/named.boot.

If you are setting up your system as a primary master server, you need to supply the following additional information:

- The BIND host file name: the default is named.hosts.
- The BIND local host file name: the default is named.local.
- The BIND reverse local host file name: the default is named.rev.
- The BIND cache file name: the default is named.ca.

After obtaining the necessary information, bindsetup creates the appropriate boot file for your BIND server. This file is /etc/named.boot by default.

If the /etc/svcorder file does not exist, bindsetup creates it and places two entries in it. One entry is for the BIND service and the other is for the local service. The local entry allows your system to perform local host and address resolution with the /etc/hosts file in the event that the BIND servers are down.

If the svcorder file already exists, the bindsetup commands reminds you to review the file and to be sure that the services are listed in the proper order. You can edit the /etc/svcorder file after the bindsetup command completes. Be sure that you have an entry in the svcorder file for local. See Section 2.4.4 and svcorder(5) in the ULTRIX Reference Pages for further information about the svcorder file.

Note

If the bindsetup command fails for any reason, be sure to check the /etc/svcorder file. If the BIND service is not properly set up, be sure that there is no bind entry in this file.

The bindsetup command edits the /etc/rc.local file. The /bin/hostname entry is changed to reflect the full BIND name, such as changing chicago to chicago.cities.dec.com. In addition, bindsetup places an entry for the named daemon before the local daemons such as sendmail. The named entry goes either before or after any YP entries, but before any NFS entries.

Finally, the bindsetup command executes the hostname command, using the new BIND hostname. The new host name is placed in the /etc/hosts file for local host name and address resolution.

2.3.2 Edit the Domain Data Files

When the bindsetup command has completed, it lists the files you need to edit manually. The default files are:

- /etc/namedb/named.hosts
- /etc/namedb/named.rev
- /etc/namedb/named.local
- /etc/namedb/named.ca
- /etc/svcorder

See Sections 2.4.4 and 2.5 for information about editing the files.

2.3.3 Reboot the System

After you have modified the files, you should reboot the system. The following command shuts down the system and reboots it immediately:

```
# /etc/shutdown -r now
```

See shutdown(8) in the ULTRIX Reference Pages for further information about shutting down the system.

Note

If you do not want to reboot your system, you can type the following command to start the named daemon and thus start the BIND service running:

```
# /usr/etc/named /etc/named.boot &
```

Be advised, however, that not all systems will necessarily know about your system's new host name. Therefore, it is best to reboot the system, if possible.

The named daemon places its process number in the file /etc/named.pid. In the event that you need to send a signal to the named process, this is where you can find its process identification (pid) number.

2.4 Setting Up a BIND Server Manually

BIND servers make use of several configuration files. To set up your system as a BIND server, the first task is to set up the BIND environment by editing the BIND configuration files. After the BIND environment has been established, you start the BIND service by rebooting the system as described in Section 2.3.3. Here are the steps:

1. Edit the boot file

2. Edit the domain data files
3. Set the host name in the hosts and rc.local files
4. Edit the svcorder file
5. Reboot the System

2.4.1 Edit the Boot File

When the named daemon starts running, it reads the boot file. This file tells the server what type of server it is, which zone it has authority for, and where to get its initial data. The default path and name for the boot file is /etc/named.boot.

Note

You can change the name of the boot file and the path by specifying the full pathname on the command line when you start the named daemon.

The boot file named.boot has two types of entries. One entry specifies the directory, and the other specifies the type of server. Here is the format of the directory entry:

directory *directory*

This entry specifies the directory where the data files reside for the named daemon to read at start time. (The directory should be large enough to hold a core dump, should one occur.) This is especially important if there are include files with relative path names called by $include. (See Section 2.5.1 for information about the include data file entry.) The default directory is /etc/namedb. For example:

directory /etc/namedb

The directory entry allows you to state just the file name in subsequent entries in the boot file. If you do not have a directory entry in the boot file, you must explicitly state the full pathname for each file name in each entry.

Here is the format for the second type of boot file entry:

type domain/zone source data file/IP addr refresh

The first field is the type variable, which qualifies the remainder of the line by specifying the type of server. The choices are primary, secondary, cache, forwarder, or slave. The second field specifies the domain or zone for which the server has authority. The third field specifies the name of the file from which the data is to be read. The fourth optional field is

specifies the interval, in seconds, for refreshing the data file. The exact content of each field depends upon the type of server:

primary

For a primary server, the first field specifies that the server is primary for the zone stated in the second field. The third field specifies the name of the file from which data is read. Thus, for a primary server in a zone called cities.dec.com with a data file called /etc/namedb/namedb.hosts, and a cache file that is refreshed every 3600 seconds (one hour), here are the proper entries:

```
directory   /etc/namedb
;type       domain/zone      data file/IP addr    refresh
primary     cities.dec.com   named.hosts
cache       .                named.ca             3600
```

Note that the entry beginning with a semicolon (;) is a comment.

secondary

The entry for a secondary server is similar to that of the primary server. The exceptions are in the first and third fields. The first field states that the server is secondary, the second field specifies the name of the zone, and the third field contains the IP addresses for each of the primary servers of the zone. For example:

```
directory   /etc/namedb
;type       domain/zone      data file                          refresh
secondary   cities.dec.com   128.11.22.33 128.11.22.44          cities.dec.com.db
cache                        named.ca
```

The data file cities.dec.com.db contains the information accumulated in the secondary server's cache. Because this file is explicitly specified, there is no need to specify a refresh time for the cache entry. If the primary server is down when a secondary server boots, the secondary server uses this file to load its cache.

Secondary servers obtain their information across the network from the listed primary servers. In this example, the two IP addresses are 128.11.22.33 and 128.11.22.44. Each primary server is tried in the order listed until the secondary server successfully obtains the information it needs, or the list is exhausted. Each server needs an entry similar to the following in its boot file:

```
primary   0.0.127-in-addr.arpa     /etc/named.local
```

This entry provides address to hostname translation for the local host.

Note

If no primary server is running when a secondary server is booted, the secondary server will not be able to load its cache. Therefore, when the server comes to multiuser mode, the BIND service will not be able to resolve all addresses, especially those in the local domain that require data from the primary server. It will, however, keep trying to resolve all queries until a primary server comes on line.

cache

The absence of an entry specifying the type of authority, or server, such as secondary or primary, designates a caching server.

All servers, however, need an entry similar to the following:

```
directory   /etc/namedb
;type       domain/zone    data file    refresh
cache       .              named.ca     3600
```

This entry primes the server's cache and designates the cache save file. In this example, the cache save file is /etc/namedb/named.ca. The optional time interval specifies how often the BIND service cache will be dumped into the cache save file. In this example, the cache is refreshed once every 3600 seconds (once per hour). The dump frequency should not be more often than once per hour. If the frequency is 0 or not specified, the cache is never dumped. In this situation, all cache files listed are read each time the system boots.

The cache entry should be the last entry in the boot file. Each time the system is brought to multiuser mode, any values that are still valid (based on the time-to-live field) are reinstated in the cache and the IP addresses of the root BIND servers in the cache files are always used.

The following shows sample entries for a caching only server:

```
;type          domain/zone              data file      refresh
primary       0.0.127-in-addr.arpa     named.local
cache         .                        named.ca       3600
```

forwarder

A forwarding server is always a primary or secondary server, but is designated a forwarder by a slave server. See the following information on slave servers in order to understand how a slave server designates which master servers will be its forwarder.

slave

Slave servers need to access forwarders in order to provide answers to resolver queries. Here are the necessary entries for a slave server:

```
slave
directory     /etc/namedb
;type         domain/zone              data file      refresh
forwarders    128.32.3.55  128.32.4.66
primary       0.0.127-in-addr.arpa     named.local
cache         .                        named.ca       3600
```

The forwarders entry lists the IP addresses of all the forwarders on the local network. In this example there are two forwarders on the network. Their addresses are 128.32.3.55 and 128.32.4.66.

The slave server will query each of the forwarders in the order they are specified until the list is exhausted or the answer is found.

2.4.2 Edit the Domain Data Files

There are four standard files for specifying the data for a domain. These are named.ca, named.local, named.hosts, and named.rev. These files use the standard format described in Section 2.5. Examples of each of these files are shown in Appendix A. The boot file specifies the location of these files, which is usually /etc/namedb.

named.ca

This file identifies the authoritative server for the zone. By default, the named.ca file contains the necessary entries for the root servers and does not need to be edited.

The format of the named.ca file follows the standard described in Section 2.5.

named.local — This file specifies the address for the local loopback interface, and is typically expressed as localhost with the network address 127.0.0.1.

named.hosts — This file contains the host and address information for all the systems in the zone.

named.rev — This file specifies the in-addr.arpa domain, which allows reverse address to name mapping. This special domain was formed to allow inverse mapping because IP host addresses do not fall within domain boundaries. The in-addr.arpa domain has four labels preceding it, which correspond to the four octets of an IP address. You must specify all four octets even if an octet is zero. For example, the IP address 128.32.0.4 is located in the domain 4.0.32.128.in-addr.arpa. This address reversal is awkward to read but allows for the natural grouping of hosts in a network. See the Guide to Networking for further information about IP addresses.

2.4.3 Set the Host Name in the hosts and rc.local Files

You need to change the host name of your system to the fully qualified BIND name. For example, if your system's name is miami and your BIND domain is cities.dec.com, you need to change your system's name to miami.cities.dec.com in both the /etc/hosts and /etc/rc.local files. Here is an example of a proper hosts file entry:

```
128.11.22.33   miami.cities.dec.com
```

Here is an example of a proper rc.local file entry:

```
/bin/hostname miami.cities.dec.com
```

2.4.4 Edit the svcorder File

The gethostbyname() library call can detect if the BIND service is selected. If the BIND service is not selected, gethostbyname() checks the /etc/svcorder file to see if there is another host lookup service to use to resolve the host name and address information.

Since the services are accessed in the order they appear in the service order file /etc/svcorder, you need to be sure that the svcorder file has its services listed in the proper order for your site. For example, assume the svcorder file has the following entries:

```
bind
yp
local
```

In this example, the BIND service is first used to resolve a query. If the BIND service fails, the YP service is used. If this fails, the local /etc/hosts file is used. Because the process of reading the /etc/hosts file is slow, it may be best for you to place the local entry last in the /etc/svcorder file. Of course, if the majority of operations on your site concern the localhost entry of the /etc/hosts file, it may be best for you to list the local entry first in the /etc/svcorder file. This would save quite a bit of traffic on the network and ease the load of your BIND server. It is cheaper to find the first or second line of the /etc/hosts file than it is to go over the IP network or through the socket software. See svcorder(5) in the ULTRIX Reference Pages for further information about the svcorder file.

Note

You should place entries for your system's local host, interface addresses, and a few names and addresses of other systems on your local network in the /etc/hosts file. Then you can use the rcp command to copy files from another system while your system is in single-user mode. Do not include duplicate host names and addresses that are covered by network services, because the /etc/hosts file does not get updated automatically. You should have the localhost and loopback entries in the /etc/hosts file, in addition to the few names and addresses of other systems on your local network. See hosts(5) in the ULTRIX Reference Pages for a description of the hosts file format.

2.4.5 Reboot the System

To have the BIND service start automatically on the BIND servers each time the system is brought to multiuser mode, place an entry for the named daemon in the /etc/rc.local file. This entry should go before any local daemon entries, such as sendmail. The entry for the named daemon can go either before or after the entries for YP, but should go before any NFS entries, if they exist. The following is a typical entry:

```
# %BINDSTART% - BIND daemon

echo -n 'BIND daemon:'      > /dev/console
[ -f /usr/etc/named ] && {
     /usr/etc/named /etc/named.boot ; echo -n ' named'   >/dev/console
}
echo '.'       > /dev/console
# %BINDEND%
```

The lines beginning with a number sign (#) are comments that make the rc.local file easier to read.

The following command shuts down the system and reboots it immediately:

```
# /etc/shutdown -r now
```

See shutdown(8) in the ULTRIX Reference Pages for further information about shutting down the system.

Note

To start the BIND service without having to bring the system to single-user and then multiuser mode, type:

```
# /usr/etc/named
```

To start the BIND service with this command, the boot file name must be the default, which is /etc/named.boot.

2.5 Format of BIND File Entries

The boot file, by default called /etc/named.boot, specifies the names of the BIND data files. These data files, also known as Resource Records (RR) consist of entries that follow the formats described in this section.

Here is the general format of a BIND data file entry (RR):

name ttl addr-class entry-type entry-specific-data

The fields are as follows:

name — This is the name of the domain, for example cities.dec.com. The domain name must begin in the first column.

For some data file entries the name field is left blank. In that case the domain name is assumed to be the same as the previous entry.

A free standing period (.) refers to the current domain.

	A free standing at sign (@) denotes the current origin, thus allowing you to specify more than one domain.
	Two free standing periods (..) represent the null domain name of the root.
ttl	This is the time-to-live field, and specifies how long, in seconds, the data will be stored in the data base. If this field is left blank, the value defaults to the ttl specified in the start of authority entry. The maximum time-to-live is 99999999 seconds, or 3 years.
addr-class	This field is the address class. There are two classes. Internet addresses are of class IN. All other types of network address are of class ANY. The address class of all data file entries in a particular zone must be the same. Therefore, only the first entry in a zone need specify the addr-class field.
entry-type	This field states the resource record type, for example SOA, A, and so forth.
entry-specific-data	All fields after the entry-type field vary for each type of date file entry (resource record).

The case is preserved in name and data fields when loaded into the BIND server. All comparisons and lookups using the BIND service are performed case insensitive.

The following characters have special meanings in BIND data file entries:

\x	A backslash (\) escapes the next non-digit (*x*) character so that the character's special meaning does not apply. For example, you could use . to place a period character in a label.
\nnn	A backslash denotes the octet corresponding to the decimal number represented by *nnn*. The resulting octet is assumed to be text and is not checked for special meaning.
( )	Parentheses group data that cross a line. In effect, line terminations are not recognized within parentheses.
;	A semicolon starts a comment, causing the rest of the line to be ignored.
*	An asterisk signifies wildcarding.

Most BIND data file entries have the current domain appended to their names if they are not terminated by a period (.). This is useful for appending the current domain name to the data, such as system names, but could cause problems when you do not want this to happen. Consequently, if the name is not in the domain for which you are creating the data file, end the name with a period.

These are the types of entries that data files (resource records) can have:

- $include
- $origin
- SOA – start of authority
- NS – name server
- A – address
- HINFO – host information
- WKS – well know services
- CNAME – canonical name
- PTR – domain name pointer
- MB – mail box
- MR – mail rename
- MINFO – mailbox information
- MG – mail group
- MX – mail exchanger

The following sections describe each of these entries and the formats they take.

2.5.1 The include Data File Entry

An include entry is similar to a header file in the C programming language. This feature is particularly useful for separating different types of data into multiple files. An include entry begins with $include in the first column, and is followed by the name of the file to be included. For example:

```
$include /etc/named/data/mailboxes
```

This entry requests the BIND service to load the data file /etc/named/data/mailboxes.

The $include entry loads data files into the local zone and acts as a data file organizer. For example, you can use $include entries to separate mail from host information.

2.5.2 The origin Data File Entry

An origin entry changes the origin in a data file. This feature is particularly useful for putting more than one domain in a data file. An origin entry begins with $origin in the first column, followed by a domain origin. For example:

```
$origin state.dec.com.
```

This entry includes the domain state.dec.com in the data file. As a result, the BIND service can provide information about the state.dec.com domain in addition to the local domain, provided your server is authoritative for the zone.

The $origin and $include entries can work together. They can also save typing and help keep the files organized. For example, assume that the following entries are in the named.rev file:

```
$origin 11.128.in-addr.arpa.
$include cities.dec.com.rev
```

The period after arpa is significant, since it signifies the complete domain name.

Assume that the cities.dec.com.rev file consists of entries similar to the following:

```
33.22 IN  PTR chicago.cities.dec.com.
```

In this situation, the complete reverse name for the host chicago is translated to be:

```
33.22.11.128.in-addr.arpa  IN PTR chicago.cities.dec.com.
```

2.5.3 The Start of Authority Data File Entry

The start of authority (SOA) entry designates the beginning of a zone. There should be no more than one SOA entry per zone. Here is the format of an SOA entry:

name ttl addr-class entry-type origin person serial# refresh retry expire min

The fields in the SOA entry have the values described in Section 2.5, with the following exceptions:

origin — This field is the name of the host on which the data file resides. This is usually a primary master server.

person — This field defines the login name and mailing address of the person responsible for the BIND service running on the local domain.

serial# — This field specifies the version number of the data file. The person editing the master files for the zone should increment the value in this field each time a change is made to the data within the file. The serial number being changed informs the secondary servers that there is new data to be obtained from the primary server. The maximum number is 9999 after the

decimal point.

The serial# field allows the BIND service to determine which of two copies of data files in a zone are more recent. Typically, the serial# field begins at one (1) and is incremented by one each time the original data file is modified. It is best to use whole integers.

refresh — This field specifies how often, in seconds, a secondary BIND server is to check with the primary server to see if it needs to update its data files. If the data files are out of date (as indicated by a mismatch of serial# fields), they are updated with the contents of the master server's files.

The minimum refresh period is 300 seconds (five minutes). If the refresh field is left blank, however, the data file is not dynamically updated.

retry — This field specifies how often in seconds, a secondary BIND server will try to refresh its data files after a refresh failure has occurred while making the check. If a BIND server attempts to refresh the files and fails, it tries to refresh them again every so many seconds, as specified in the retry field.

expire — This field specifies the upper limit, in seconds, that a secondary BIND server can use the data files in its cache before the data expires for lack of being updated, or before the BIND server checks to see if its cache needs to be updated.

min — This field specifies the default time to live, in seconds, that a data entry can exist in the event that the ttl entry is left blank.

The following is an example of an SOA entry. The first line is a comment that shows the fields:

```
;name    ttl    addr-class    entry-type    origin                          person
@               IN            SOA           utah.states.dec.com             hes.utah.states.dec.com. (
                                            1          ; serial
                                            3600       ; refresh every hr.
                                            300        ; retry every 5 min.
                                            3600000    ; expire in 100 hrs.
                                            86400 )    ; min. life is 24 hrs.
```

In this example note that the parentheses indicate to the BIND service that this is a single entry. the ttl field is left blank, indicating that the default time to live specified in the min field (86400 seconds) is being used.

The semicolons allow comments for readability. In the example, the serial field is 1, the refresh field is 3600 seconds (once per hour), the retry field is 300 seconds (once per 5 minutes), the expire field is 3,600,000 (100 hours), and the minimum field is 86400 seconds (24 hours).

2.5.4 The Name Server Data File Entry

The name server (NS) entry specifies which system is the primary master server, that is, which BIND server is responsible for the domain. There should be only one NS entry for each primary master server on the domain. Here is the format of the NS entry:

name ttl addr-class entry-type server

The fields in the NS entry have the values described in Section 2.5, with the exception of the server field. This field specifies the name of the primary master server for the domain specified in the first field.

Here is an example of an NS entry:

```
;name    ttl    addr-class    entry-type    server
                IN            NS            utah.states.dec.com.
```

In this example note that the first and second fields are left blank, thus using the domain specified in a previous entry and the ttl specified in the SOA entry.

2.5.5 The Address Data File Entry

The address (A) data file entry lists the address for a specific system. Here is the format for an A entry:

name ttl addr-class entry-type address

The fields in the A entry have the values described in Section 2.5, with the exception of the address field. This field specifies the IP address for each system. There should only be one A entry for each address of a given system.

Here is an example of two A entries:

```
;name                    ttl    addr-class    entry-type    address
miami.cities.dec.com.           IN            A             128.11.22.44
                                IN            A             128.11.22.33
```

In this example note that the first entry has left the ttl field blank, thus using the default ttl specified in the SOA entry. The second entry has

in the previous entry and the default ttl specified in the SOA entry. In this example, the host miami.cities.dec.com has two IP addresses.

2.5.6 The Host Information Data File Entry

The host information (HINFO) data file entry is for host specific information. This entry lists the hardware and operating system that are running at the specified host system. Only a single space separates the name of the hardware from the operating system information. Thus, if you need to use spaces as part of a host or operating system name, you must place the name in quotes. In addition, there can be no more than one HINFO entry for each host on the domain. Here is the HINFO entry format:

host ttl addr-class entry-type hardware opsys

The fields in the HINFO entry have the values described in Section 2.5, with the following exceptions:

host — This field specifies the host name. If the host is in the current domain, you only need to specify the host, say chicago, for example. If the host is in a different domain, you must specify the full BIND name, for example: utah.state.dec.com.. Be sure to include the period (.) at the end of the host name. This indicates the fully qualified BIND name.

hardware — This field specifies the type of CPU, for example, a VAX 8800 processor.

opsys — This field specifies the type of operating system running on the specified host and should be ULTRIX for the ULTRIX operating system.

Here is an example of a HINFO entry:

;name	ttl	addr-class	entry-type	hardware	opsys
ohio.state.dec.com.		IN	HINFO	X-11/780	ULTRIX

In this example, note that the second field specifying the ttl is blank, thus using the default ttl specified in the SOA entry.

2.5.7 The Well Known Services Data File Entry

The well know services (WKS) entry describes well known services supported by a particular protocol at a specified address. The services and port numbers are obtained from the list of services specified in the /etc/services file. Here is the format of a WKS entry:

name ttl addr-class entry-type address protocol services

The fields in the WKS entry have the values described in Section 2.5, with the following exceptions:

address — This field specifies the IP address for each system. There should only be one WKS entry for each protocol at each address.

protocol — This field specifies the protocol to be used, for example TCP or UDP.

Here is an example of two WKS entries:

```
;name    ttl   addr-class    entry-type    address       protocol    services
                IN            WKS           128.32.0.4    UDP         who route
                IN            WKS           128.32.0.78   TCP         (echo talk
                                                                      discard sunrpc sftp
                                                                      uucp-path netstat host
                                                                      systat daytime link
                                                                      auth time ftp
                                                                      nntp whois pop
                                                                      finger smtp supdup
                                                                      domain nameserver
                                                                      chargen)
```

Note that the first and second fields of both entries in this example are blank, which indicates that they are using the domain name specified in a previous entry and the default ttl specified in the SOA entry. The services listed in the second entry are contained within parentheses and are thus interpreted as being one entry, even though they appear to be on several lines.

2.5.8 The Canonical Name Data File Entry

The canonical name (CNAME) entry specifies an alias for a canonical name. For example, if the canonical name, (also known as the full BIND name or the fully qualified name) is miami.cities.dec.com, a reasonable alias might be miami or mi.

An alias must be unique, and all other entries should be associated with the canonical name and not with the alias. Do not create an alias and then use it in other entries. Here is the format of a CNAME entry:

aliases ttl addr-class entry-type can-name

The fields in the CNAME entry have the values described in Section 2.5,

aliases — This field specifies the nickname, or alias, of the canonical name of the host.

can-name — This is the canonical name of the host. If the canonical name is a part of the current domain, then you only need to specify the host name, for example, miami. If the canonical name is for a host in another domain, you must specify the fully qualified BIND name, followed by a period (.). For example: ohio.state.dec.com.

The following example shows two CNAME entries. The first entry is for a CNAME in the current domain; the second entry is for a CNAME in another domain:

;aliases	ttl	addr-class	entry-type	can-name
to		IN	CNAME	toledo
mon		IN	CNAME	ohio.state.dec.com.

2.5.9 The Domain Name Pointer Data File Entry

The domain name pointer (PTR) entry allows special names to point to some other location in the domain. PTR names should be unique to the zone. Here is the format of a PTR entry:

rev-addr ttl addr-class entry-type realname

The fields in the PTR entry have the values described in Section 2.5, with the following exceptions:

rev-addr — This field specifies the reverse IP address of the host. You can obtain the reverse address from the /etc/namedb/named.rev file. For example, if the host's address is 128.11.22.33, the reverse address is 33.22.11.128.

realname — This is the fully qualified (canonical) BIND name of the host, for example, miami.cities.dec.com. Be sure to include the period (.) at the end of the real name if the host is not in the current domain.

Here is an example of two PTR entries:

;rev-addr	ttl	addr-class	entry-type	realname
33.22		IN	PTR	chicago
66.55.44.121		IN	PTR	mail.peace.org.

In this example, the first entry is for a host whose IP host address is 22.33 in the current domain. The specified rev.addr (33.22) is meaningful assuming that a $origin entry exists. See Section 2.5.2 for a description

of the $origin entry. If there is not a $origin entry, then the entire IP address, in reverse, must be specified.

The second entry is for a host in different domain (state.dec.com.). As a rule, you should not do this because you are putting data in your server's cache for which your server is not authoritative. PTR entries and other resource records should be for hosts in your domain, only.

The PTR entry sets up a reverse pointer for the special domain peace.org.

2.5.10 The Mailbox Data File Entry

The mailbox (MB) entry lists the system where a user wants to receive mail. Here is the format of an MB entry:

login ttl addr-class entry-type system

The fields in the MB entry have the values described in Section 2.5, with the following exceptions:

login — This field is the login name for a user. Login names must be unique for the domain.

system — This field specifies the name system where the user wants to receive mail.

Here is an example of an MB entry:

```
;login    ttl    addr-class    entry-type    system
fred             IN            MB            potsdam.cities.dec.com.
```

In this example note that the second field is left blank, thus using the ttl specified in the SOA entry. Consequently, the user fred will have mail delivered to the host named potsdam in the domain cities.dec.com.

2.5.11 The Mail Rename Data File Entry

The mail rename (MR) entry lists aliases for a specific user. Here is the format of an MR entry:

alias ttl addr-class entry-type login

The fields in the MR entry have the values described in Section 2.5, with the following exceptions:

alias — This field lists the nicknames for the specified user. The alias must be unique to the domain.

login — This field is the login name for the user whose alias is being established. There should also be a corresponding MB entry for the specified login name.

Login names must be unique for the domain.

Here is an example of an MR entry:

```
;alias      ttl   addr-class   entry-type   login
lady              IN           MR           diana
princess          IN           MR           diana
```

This example shows how to set up the aliases lady and princess for a user whose login name is diana. Note that the second field is left blank, thus using the ttl specified in the SOA entry.

2.5.12 The Mailbox Information Data File Entry

The mailbox information (MINFO) entry creates a mail group for a mailing list. The MINFO entry is usually associated with a mail group (MG) entry, but can also be used with a mailbox (MB) entry. Here is the format of a MINFO entry:

mailbox ttl addr-class entry-type requests maintainer

The fields in the MINFO entry have the values described in Section 2.5, with the following exceptions:

mailbox — This field specifies the name of the mailbox, and is usually BIND.

requests — This field specifies the name where users should send mail relating to the BIND service or mail. For example, a user might want to send a mail message requesting that an alias be set up.

maintainer — This field contains the login name of the person who should receive mail error messages. This is particularly useful when an error in member's names should be reported to a person other than the sender.

Here is an example of a MINFO entry:

```
;mailbox   ttl   addr-class   entry-type   requests       maintainer
BIND              IN           MINFO        BIND-REQUEST   fred@ miami.cities.dec.com.
```

In this example, note that the second field is left blank, thus using the ttl specified in the SOA entry.

2.5.13 The Mail Group Data File Entry

The mail group entry specifies the members of a mail group. The MG entry is usually used with a MINFO entry. Here is the format of an MG entry:

group ttl addr-class entry-type member

The fields in the MG entry have the values described in Section 2.5, with the following exceptions:

group — This field specifies the name of the mail group, for example, users or marketing.

member — This field specifies the login name and the domain of the user to be included in the mail group.

Here is an example of a MINFO entry and several MG entries:

;group	ttl	addr-class	entry-type	requests	member
fun		IN	MINFO	BIND-REQUEST	fred@ miami.cities.dec.com.
		IN	MG		john@ miami.cities.dec.com.
		IN	MG		amy@ miami.cities.dec.com.

In this example, note that the second field for all three entries is left blank, thus using the ttl specified in the SOA entry. In addition, if mail is sent to the mail group fun, fred, john, and amy receive it.

2.5.14 The Mail Exchanger Data File Entry

The mail exchanger (MX) entry specifies a system in the local domain (called a gateway) that knows how to deliver mail to a system that may not be directly connected to the local network. Consequently, the MX entry is useful for systems outside your local network that want to send mail to a user on one of your network's hosts.

You can also use the MX entry to list some of the hosts in the /etc/hosts file so that they do not appear to other systems using the BIND service.

Here is the format of an MX entry:

system ttl addr-class entry-type pref-value gateway

The fields in the MX entry have the values described in Section 2.5, with the following exceptions:

system — This field specifies the name of the system where mail is to be sent.

pref-value — This field specifies the order a mailer should follow when there is more than one way to deliver mail to a given system.

gateway This field contains the name of the gateway system, that is, the system that can deliver mail to the destination system on another network.

Here is an example of two MX entries:

;system	ttl	addr-class	entry-type	pref-value	gateway
tampa.cities.dec.com		IN	MX	0	seismo.cs.au.
*.folks.dec.com		IN	MX	0	relay.cs.net.

In this example, all mail destined for the domain folks.dec.com, regardless of the host name, is sent by route of the relay.cs.net. host. In addition, note that the second field in both entries is left blank, thus using the ttl specified in the SOA entry. The second entry uses an asterisk, which is a wildcard.

Managing and Using the BIND Service 3

This chapter provides the background information required for maintaining and using the BIND service. Included is a description of the domain administrator and the technical and zone contacts, as well as the duties of each.

This chapter describes how to register your site with the public networks and where to find additional information about the BIND name server.

Finally, this chapter provides a brief tutorial on how to make use of the BIND service for obtaining host names and IP addresses. In addition, the nslookup and nsquery commands are also introduced.

3.1 Maintaining the Domain

BIND domains are administrative entities that provide decentralized management of host names and addresses. The domain naming scheme is distributed and hierarchical. The Network Information Center (NIC) maintains the zone files of the root domain BIND server. The NIC also maintains the top-level domains arpa, com, edu, gov, mil, and org, plus a number of country domains. In addition, the NIC registers first and second-level domains.

The domain administrator (DA) administers each local domain with the help of the technical and zone contacts. These roles are described in the following sections.

3.1.1 Domain Administrator Role

Typically, each BIND domain has a domain administrator (DA), who is responsible for coordinating and managing the domain. The DA registers a second-level or lower domain by interacting with the DA in the next higher level domain. For information on finding the names of the DA contacts, see section 3.5.3.

The DA duties include:

- Understanding the concepts and procedures of the BIND service

- Ensuring that the service is reliable
- Ensuring that the BIND data is current
- Taking prompt action when necessary, for example if protocols are violated or other serious misbehavior
- Controlling the assignments of the host and domain names

The DA furnishes users with access to names and name-related information both inside and outside the local domain. In addition, the DA works closely with the domain technical and zone contacts for the domain.

3.1.2 The Technical and Zone Contact

Typically, the technical and zone contact is concerned with the technical aspects of maintaining the BIND server and resolver software and the data files. The technical and zone contact keeps the BIND server running and interacts with technical people in other domains and zones to solve problems affecting the local domain.

A zone consists of those contiguous parts of the domain tree for which a domain server has complete information and over which it has authority. A BIND server can be the authority for several zones.

3.2 Naming Domains and Hosts

The NIC makes domain name assignments on a first-come, first-served basis. The NIC only registers domains under the top-level domains, not individual hosts. This allows administration and data maintenance to be delegated down the hierarchical tree.

A domain is identified by a domain name, and consists of that part of the domain name space that is at or below the domain name. A domain is a subdomain of another larger domain, if it is contained within that domain. That is, if a domain's name ends with the containing domain's name, of which it is a subdomain. For example, A.B.C.D is a subdomain of B.C.D, C.D, D, and the root domain (.).

There are two types of names:

- The fully qualified name represents the complete domain name. This is also known as the absolute or canonical name. For example:

  ```
  chicago.cities.dec.com.
  ```

- Relative names represent the starting name (label) of an absolute domain name. Relative names are incomplete, but are completed by the BIND service, using knowledge of the local domain, for example:

```
chicago
```

Relative host names such as chicago are automatically expanded to the fully qualified domain name (chicago.cities.dec.com) when given in a typical command.

Domain and host names must begin with a letter, end with a letter or digit, and have only letters, digits, or hyphens as internal characters. Although the names can be up to 64 characters, it is best to choose names that are 12 characters or fewer because the canonical (fully qualified) domain names are easier to keep track of if they are short. The sum of all the label octets and label lengths is limited to 255.

Note

Domain names are case insensitive. By convention, however, whenever you receive a domain name you should preserve its case.

It is up to the DA to resolve any local conflicts concerning the domain name chosen.

Note

Countries can register as top-level domains provided they name themselves after a two-letter country code listed in the international standard ISO-3166. (Appendix C lists several BIND standards.) In the event that a country code is identical to a state code that the U.S. Postal Service uses, the country can request a three-letter code.

3.3 Registering With Public Networks

Before you can set up the BIND service on your system, your system must be established on a local area network. If the BIND service for your domain is part of a public network, you should get in touch with the organization in charge of that network and request the appropriate domain registration form. Even if your site belongs to more than one network, you should register your site with only one. The following sections describe how to contact these networks:

- DARPA Internet network (ARPANET)
- CSNET
- BITNET

3.3.1 Contacting the DARPA Internet Network

If your system is on the DARPA (Defense Advanced Research Projects Agency) Internet network (also known as the ARPANET), contact the following organization:

```
hostmaster@sri-nic.arpa
```

The people there will provide you with information about setting up a BIND domain.

You can also request to be placed on the BIND mailing list. This mailing list is for people running BIND on the DARPA Internet network who want to discuss future designs, operational problems, and other related topics. Here is the address:

```
bind-request@ucbarpa.berkeley.EDU
```

3.3.2 Contacting the CSNET

If your site's domain name is not already registered with the CSNET (Computer Science Network), contact the CSNET Coordination and Information Center (CIC). They will send you an application and provide you with information and technical advice about setting up a domain.

If your site's domain name is already registered with the CIC, you should keep the CIC informed of how you want your site's mail routed. In general, the CSNET relay prefers to send mail by CSNET, rather than by the BITNET or the ARPANET. If your site is on more than one network, the CSNET relay might not be the preferred route.

You can contact the CIC at the following electronic mail address:

```
cic@sh.cs.net
```

Or, you can reach the CIC hotline at this phone number:

```
(617) 873-2777
```

3.3.3 Contacting the BITNET

Some colleges and universities are on the BITNET network. This network is reserved for students, faculty, and scholars who want to communicate on a common network. BITNET stands for: "Because It's Time Network."

If your site is on the BITNET and you want to set up a domain, contact the following address or phone number for information:

BITNET Network Information Center (BITNIC)
Educom
Bitnet Network Information Center
P.O. Box 364
Princeton, NJ 08540
(609) 520-3340

For general information, send electronic mail to:

bitserve@CUNYVM

For general inquiries, send electronic mail to:

info%bitnic.bitnet@CUNYVM.CUNY.EDU

3.4 Updating BIND Data Files

Occasionally you may need to update the BIND data files. For example, you may need to add a host to the data files. To update the data file – for example to add a host – here are the steps:

1. Be sure the minimum refresh time on the secondary servers is at least five minutes (300 seconds).
2. Edit the appropriate data files on the primary server. If you are adding a host name, you typically need to edit the /etc/namedb/named.rev file and any other files with an SOA record for your domain.
3. Increment the serial# field of the SOA entry in the appropriate data files on the primary server. For example, if you are adding a host name, you probably need to increment the SOA entry for the domain in the /etc/namedb/named.rev file, as well as any other data base files you may have set up for host names and addresses.

   If you neglect to change the serial# field, the secondary servers will not be aware of the modified data when they check their serial# fields against the primary server's to see if they need to refresh their data files.

   The serial# field typically starts at one (1) and is incremented by one each time the data is modified.
4. Tell the primary server to reload the data base by sending the -HUP signal to the named daemon as follows:

```
# kill -HUP `cat /etc/named.pid`
```

3.5 Obtaining Host Name and IP Address Information

There are several ways that you can obtain information about host name and IP addresses from a system using the BIND service. The following sections provide an introduction to these commands:

- nslookup
- nsquery
- whois

3.5.1 The nslookup Command

One way to obtain information about host name and IP addresses is with the nslookup command. With this command, you can non-interactively and interactively query the BIND service for information about hosts on the local, as well as remote, domains. You can also find information about BIND resource records such as MX, NS, and so forth.

Here is the format for a non-interactive query with the nslookup command:

nslookup *hostname*

A good way to learn how to use the nslookup options is to experiment with it. Appendix D provides a sample interactive session with the nslookup command. For further information, see nslookup(1) in the ULTRIX Reference Pages.

To find out MX information, you need to supply the nslookup command with a bogus host name and a valid domain name. For example, to get an answer to the question, "who takes mail for the domain mit.edu?", you could type the following:

```
# nslookup
Default Server:  oops.cities.dec.com
Address:  128.54.54.1

> set querytype=mx
> find MX.mit.edu
Server:  oops.cities.dec.com
Address:  128.54.54.1

findMX.mit.edu.cities.dec.com      preference = 51, mail exchanger = noun.cities.dec.com
findMX.mit.edu.cities.dec.com      preference = 50, mail exchanger = wepel.cities.dec.com
noun.cities.dec.com        inet address = 128.54.54.79
wepel.cities.dec.com       inet address = 128.54.54.93
```

(continued on next page)

```
> <CTRL/d>
#
```

In this example, the host name MX.mit.edu. is bogus, but the domain mit.edu. is real. See Appendix D for further examples of nslookup command sessions.

3.5.2 The nsquery Command

The nsquery command provides a quick, non-interactive method for obtaining host names, aliases, and IP addresses. The following example shows how to get the host name, alias, and IP address for a host called chicago:

```
# nsquery chicago
Name: chicago
Address: 128.11.22.333
Aliases: c ch
```

See nsquery(1) in the ULTRIX Reference Pages for further information.

3.5.3 The NIC whois Service

The NIC whois service allows you to verify the following information:

- The name and address of the organization responsible for the domain
- The name of the domain
- The domain's administrative and technical and zone contacts
- The host names and network addresses of sites providing the BIND service for the domain

To use the NIC whois service to find information about a domain named roads, send mail specifying the whois command and the domain in question in the subject header:

```
# mail service@sri-nic.arpa
Subject: whois domain rice.edu
CTRL/d
Cc:
Null message body; hope that's ok
```

Here is a sample response:

```
From SERVICE-REPLY@SRI-NIC.ARPA Thu Jun  2 17:58:38 1988
Received: from chicago.cities.dec.com (chicago.ARPA) by paris.cities.dec.com (1.2/dv.5.yp)
        id AA17498; Thu, 2 Jun 88 17:57:20 edt
Received: by chicago.cities.dec.com (5.57/v2.4)
        id AA03640; Thu, 2 Jun 88 17:56:49 EDT
Message-Id: <8806022156.AA03640@chicago.cities.dec.com>
```

From: NIC Mail Service <SERVICE-REPLY@SRI-NIC.ARPA>
To: jane@chicago (h jane ramburg-crane)
Subject: Re: whois domain rice.edu
Status: RO

Rice California (RICE-DOM)
Advanced Studies and Research
Houston, TX 77001

Domain Name: RICE.EDU

Administrative Contact:
Kennedy, Ken (KK28) Kennedy@LLL-CRG.ARPA (713) 527-4834
Technical Contact, Zone Contact:
Riffle, Vicky R. (VVR) rif@RICE.EDU
(713) 527-8101 ext 3844

Domain servers in listed order:

RICE.EDU	128.42.5.1
PENDRAGON.CS.PURDUE.EDU	128.10.2.5

3.6 Obtaining Further Information about the BIND Service

The NIC has several online documents which you can access to obtain further information about the BIND service. Some of these documents are:

NETINFO:DOMAINS.TXT
: This file consists of an informational table of the top-level domains and their root servers. This file is updated each time a new top-level domain is approved.

NETINFO:DOMAIN-INFO.TXT
: This file contains a concise list of all top-level and second-level domain names registered with the NIC. This file is updated monthly.

NETINFO:DOMAIN-CONTACTS.TXT
: This file lists each of the top-level and second-level domains, and includes the administrative, technical and zone contacts for each as well.

NETINFO:DOMAIN-TEMPLATE.TXT
: This file contains the questionnaire to be completed before registering a top-level or second-level domain. A copy of this document is in Appendix B.

You can use the ftp command to transfer copies of the online documents from SRI-NIC.ARPA. Appendix B provides a sample ftp session. Or, you can open a TCP or UDP connection to the NIC host name server, port 101 on SRI-NIC.ARPA. From there, you can invoke the command ALL-DOM. Appendix C lists several other articles and RFCs which may be of interest to you.

For further information about the BIND service, you can do the following:

- Send electronic mail to:

  HOSTMASTER@SRI-NIC.ARPA

- Call the toll-free NIC hotline at:

  (800) 235-3155

Troubleshooting the BIND Service 4

This chapter contains guidelines for troubleshooting the BIND service, as well as information for starting, controlling and debugging the named daemon.

If the BIND service fails to work properly, the cause is typically one of the following:

- The data files are not set up properly
- The BIND service cannot access the root servers

The following files and daemon are crucial to the proper working of the BIND service:

- The standard domain server data files are located in the directory /etc/namedb, and are usually named.boot, named.rev, named.hosts, named.local, and named.ca
- The /etc/svcorder file
- The /etc/rc.local file
- The /etc/resolv.conf file (for BIND clients, only)
- The named daemon (for BIND servers, only)

The following sections describe these files and the daemon in greater detail.

4.1 Reviewing the Domain Data Files

This section offers some suggestions of what to do in the event that the BIND service is not working properly.

First, be sure that the domain data files are set up correctly. Specifically, be sure that the following are correct:

- The local host in the boot file and cache files
- The in-addr domain in the boot file and any other data base files
- The reverse arp IP addresses
- The host names are in the correct domain

In addition, be sure tht there is only one reverse address per host in the domain.

If the preceding criteria is correct and you are still experiencing problems, you should continue troubleshooting the BIND service as described in the rest of this chapter.

For information about the domain data files, see Chapter 2. For examples of domain data files, see Appendix A.

4.2 Reviewing the /etc/rc.local File

Make sure that the host name is set to the fully qualified (canonical) BIND name in the the /etc/rc.local file. Be sure that an entry similar to the following one exists in the /etc/rc.local file. Here is the format for the entry:

```
/bin/hostname    host.domain
```

For example, here is the appropriate entry for a system named chicago in the domain cities.dec.com:

```
/bin/hostname chicago.cities.dec.com
```

The following entry starts the domain name server each time the system goes to multiuser mode:

```
# BINDSTART
echo -n 'BIND daemon:'      > /dev/console
[ -f /usr/etc/named ] && {
        /usr/etc/named /etc/named.boot; echo -n ' named' > /dev/console
}
echo '.'
# BINDEND
```

This entry belongs either before or after any YP entries, but before any NFS entries, if they exist. If YP and NFS entries do not exist in the /etc/rc.local file, the named entry belongs before the local daemons such as sendmail.

Note

Do not run the named daemon directly from inetd. This causes continual restarts of the name server and defeats the purpose of having a cache.

4.3 Reviewing the Resolver File

Make sure that the resolver file /etc/resolv.conf is accurate. It should contain at least one master server. See chapter 2 for information about the resolver file.

4.4 Reviewing the Debug Files

If after reviewing the rc.local file and the resolver file you are still experiencing problems, there are several other files to help you troubleshoot the BIND service further. These files are:

- /var/spool/mqueue/syslog
- /var/tmp/named_dump.db
- /var/tmp/named.run
- /tmp/named.stat

This section provides general information about the debug files and explains how to use them to troubleshoot the BIND service.

4.4.1 The syslog File

If the BIND service cannot access the root servers, it cannot resolve queries about hosts in other domains. One way to determine if the root servers are accessible is to look in the /var/spool/mqueue/syslog file. The key phrase is:

```
root hints too low
```

This key phrase indicates how many of the available root servers are actually accessible to your system. The minimum threshold is two, and the maximum is the number of root servers available at their various addresses (currently 10). If the number of root hints is too low, either the BIND files are not configured properly or one or more of the links to the root servers is down.

In addition, the named daemon may log error messages in the syslog file.

Here is a sample syslog file:

```
Jun 21 04:05:05 syslog restart
Jun 21 12:09:51 localhost: 1688 named: restarted
Jun 21 12:09:51 localhost: 1688 named: /etc/named.primaray/named.boot: No such file or directory
Jun 21 12:10:49 localhost: 1692 named: restarted
Jun 21 12:12:16 localhost: 1692 named: ..(new) named started..
Jun 21 12:17:30 localhost: 1705 sendmail: AA01705: message-id = <8806211616.AA01705@chicago.cities.dec.com>
Jun 21 12:17:31 localhost: 1705 sendmail: AA01705: from = jane, size = 243562, class = 0
Jun 21 12:17:49 localhost: 1707 sendmail: AA01705: to = jane@orlando, delay = 00:01:18, stat = Sent
Jun 21 14:50:37 localhost: 1692 named: reloading nameserver
Jun 21 14:50:45 localhost: 1692 named: 0 root hints... (too low)
Jun 21 15:20:46 localhost: 1692 named: 0 root hints... (too low)
Jun 21 15:50:46 localhost: 1692 named: 0 root hints... (too low)
Jun 21 15:59:02 localhost: 1840 sendmail: AA01840: message-id = <8806211958.AA01840@chicago.cities.dec.com>
Jun 21 15:59:02 localhost: 1840 sendmail: AA01840: from = jane, size = 835, class = 0
Jun 21 15:59:12 localhost: 1842 sendmail: AA01840: to = jane@tempe, delay = 00:00:20, stat = Sent
```

4.4.2 The named_dump.db File

If you send the named daemon a signal to dump the data base, a copy of the data base is dumped in the file /var/tmp/named_dump.db. Here is how to send the signal:

```
# kill -INT `cat /etc/named.pid`
```

By examining the resulting named_dump.db file you can determine whether any of the BIND data files are set up incorrectly. Here are some things to look for:

- Is the local loopback correct?
- Is the inaddr entry correct?
- Is the local host set up correctly?
- Are the reverse arp IP addresses correct?
- Is there a reverse address for each host?
- Are the host names in the correct domain?

Appendix A lists a sample named_dump.db file for a BIND server whose data files are correct.

4.4.3 The named.run File

If you turn on debugging, the results are logged in the file /var/tmp/named.run. There are up to 11 debug levels. Typically, however, you should debug the named daemon at level five. Then, from glancing at the named.run file you should be able to get an idea of whether the BIND

named.run file indicates a connection to a root server:

```
TCP connected
```

The following lines in the named.run file indicate a poorly running system:

- Several QUESTIONS, but no ANSWERS
- Several iterations of findns, which attempt to find a name server
- Several iterations of schedretry, which schedule another attempt to access a root server

Appendix A lists two named.run files: one for a system that is running the BIND service properly, the other for a system that is not.

4.4.4 The named.stats File

The /var/tmp/named.stats file lists the statistics for the BIND service. From this file you can see how much activity is being generated for the BIND server. To generate this file, send a signal to the named daemon. For example:

```
# kill -IOT `cat /etc/named.pid`
```

See Section 3.7.6 for more information about how to send signals to the named daemon.

After the named daemon is running with the -IOT signal, it generates the named.stats file. Here is an example of this file:

```
###  Thu June 21 15:05:09 1988
3389 time since boot (secs)
3389 time since reset (secs)
72   input packets
72   output packets
69   queries
0    iqueries
0    duplicate queries
3    responses
0    duplicate responses
69   OK answers
0    FAIL answers
0    FORMERR answers
2    system queries
1    prime cache calls
1    check_ns calls
0    bad responses dropped
0    martian responses
0    Unknown query types
60   A querys
4    NS querys
2    MX querys
3    ANY querys
```

The named.stats file may have entries for martian responses. A martian response indicates a query response from a host that is unknown to the server.

4.5 Obtaining the named Process ID

When the named daemon starts running, it places its process identification number (pid) in the file /etc/named.pid. This feature is useful for programs that need to send signals to the named daemon.

4.6 Sending Signals to the named Daemon

You can send several signals to the running named process without having to restart it.

You can also find the named process ID (pid) by using the ps command or by using the cat command with /etc/named.pid.

The signals you can send to named are as follows:

SIGHUP This signal causes the named process to read the boot file and reload the data base. However, all previously cached data is

and you want named's internal data base to reflect the change.

SIGINT — This signal dumps the current data base and cache to the file /var/tmp/named_dump.db. This can give you an indication of whether the data base was loaded correctly.

USR1 — This signal turns on debugging. Each subsequent USR1 signal increments the debug level. A good rule of thumb is to increment the debug level to five (this is accomplished by issuing the signal five times). The output is appended to the file /var/tmp/named.run.

After turning on debugging, you can try using the nslookup command and watch the debug trace. Appendix A has an example of two named.run files; one is from a system with the BIND service running properly, the other is from a system that cannot reach any of the root servers.

Here is an example of how to send the USR1 signal to the named daemon:

```
# kill -USR1 `cat /etc/named.pid`
```

You can start the named daemon in debug mode by typing the following command:

```
# /usr/etc/named named.boot -d5 &
```

This command starts the named daemon and sets the debug level to five.

USR2 — This signal turns off debugging completely.

```
# kill -USR2 `cat /etc/named.pid`
```

KILL — This signal terminates the named process. To stop the BIND service from running in the future, comment out the bind (or BIND) entry in the /etc/svcorder file by placing a number sign (#) in the first column of the BIND entry.

Appendix A

This appendix provides sample BIND files to help you understand, maintain, and troubleshoot the BIND service. The following files are listed in this appendix:

- The named.boot file
- The named.ca file
- The named.local file
- The named.hosts file
- The named.rev file
- The named_dump.db file
- A healthy named.run file
- An unhealthy named.run file

Note

Sample BIND files are not provided for root servers. If you are establishing your system as a root server, you can get help from the NIC, as stated in Chapter 1.

A.1 The named.boot File

Only BIND servers need a boot file. The default name and location of the boot file is /etc/named.boot. This section provides a sample boot file for each type of BIND server: primary master, secondary master, slave, and caching. Note that each type of server needs an entry of the form:

```
primary   0.0.127-in-addr.arpa     /etc/named.local
```

This entry provides the address-to-hostname translation for the local host.

Here are the sample boot files:

- Primary Master Server Boot File

```
;
; Data file to boot a BIND primary master server.
```

```
;
; directory where all the data files are stored
directory    etc/namedb
;
; type       domain                  source host/file
primary      cities.dec.com          named.hosts
;
primary      33.22.128.in-addr.arpa  named.rev
;
primary      0.0.127.in-addr.arpa    named.local
;
; load the cache data last
cache        .                       named.ca
```

- Secondary Master Server Boot File

```
;
; Data file to boot a BIND secondary master server.
;
; directory where all the data files are stored
directory    /etc/namedb
;
; type       domain                  source host/file
secondary    cities.dec.com          128.11.22.33 128.11.22.44
;
secondary    33.22.128.in-addr.arpa  128.11.22.33 128.11.22.33
;
primary      0.0.127.in-addr.arpa    named.local
;
; load the cache data last
cache        .                       named.ca
```

- Slave Server Boot File

```
;
; Data file to boot a BIND slave server.
;
; directory where all the data files are stored
directory    /etc/namedb
;
; type       domain                  source host/file
primary      0.0.127.in-addr.arpa    named.local
;
```

```
forwarders                      128.11.22.33 128.11.22.33
; load the cache data last
cache           .               named.ca
;
```

- Caching Server Boot File

```
;
; Data file to boot a BIND caching only server.
;
; directory where all the data files are stored
directory   /etc/namedb
;
; type      domain              source host/file refresh
primary     0.0.127.in-addr.arpa  named.local
;
; load the cache data last
cache       .                   named.ca              3600
;
```

A.2 The named.ca File

Only BIND servers running the named daemon need a cache file. The default name and location of the cache file is /etc/namedb/named.ca. Here is a sample cache file:

```
; @(#)named.ca      4.2         (ULTRIX)    3/16/88
;
; Data file for initial cache data for BIND root domain servers.
;domain             ttl         addr-class    entry-type    server
;
.                   99999999    IN            NS            ns.nasa.gov.
.                   99999999    IN            NS            sri-nic.arpa.
.                   99999999    IN            NS            a.isi.edu.
.                   99999999    IN            NS            gunter-adam.arpa.
.                   99999999    IN            NS            brl-aos.arpa.
.                   99999999    IN            NS            terp.umd.edu.
.                   99999999    IN            NS            c.nyser.net.
sri-nic.arpa.       99999999    IN            A             26.0.0.73
                    99999999    IN            A             10.0.0.51
a.isi.edu.          99999999    IN            A             26.3.0.103
gunter-adam.arpa.   99999999    IN            A             26.1.0.13
brl-aos.arpa.       99999999    IN            A             192.5.25.82
```

```
                99999999    IN      A       128.20.1.2
ns.nasa.gov.    99999999    IN      A       128.102.16.10
c.nyser.net.    99999999    IN      A       128.213.5.17
terp.umd.edu.   99999999    IN      A       10.1.0.17
                99999999    IN      A       128.8.10.90
```

A.3 The named.local File

Only BIND servers need a local file. The default name and location of the local file is /etc/namedb/named.local. Here is a sample named.local file:

```
;   @(#)named.local     4.1     (ULTRIX)     1/18/88
;
; Data file for local loopback interface.
;name    ttl    addr-class    entry-type    origin
@               IN            SOA           host.domain.    sysmgr.host.domain.  (
                                            1               ; Serial
                                            3600            ; Refresh
                                            300             ; Retry
                                            3600000         ; Expire
                                            3600 )          ; Minimum
                IN            NS            host.domain.
        1       IN            PTR           localhost.
;
```

A.4 The named.hosts File

Only BIND servers need a hosts file. The default name and location of the hosts file, specified in the boot file, is /etc/namedb/named.hosts. Here is a sample named.hosts file:

```
;   @(#)named.hosts     4.1     (ULTRIX)     1/18/88
;
; Data file of hostnames in this domain.
;name          ttl    addr-class    entry-type    origin          person
@                    IN            SOA           host.domain.    sysmgr.host.domain. (
                                                                   1
                                                                   3600
                                                                   300
                                                                   3600000
                                                                   3600 )
```

```
localhost       IN      A       127.0.0.1
sri-nic.arpa    IN      A       10.0.0.51
host            IN      A       111.22.33.44
                IN      HINFO   VAXstation2000 ULTRIX
anotherhost     IN      A       111.22.33.55
                IN      HINFO   VAXstationII ULTRIX
onemorehost     IN      A       111.22.33.66
                IN      HINFO   VAX8800 ULTRIX
```

A.5 The named.rev File

Only BIND servers need a reverse hosts file. The default name and location of the reverse hosts file is /etc/namedb/named.rev. Here is a sample named.rev file:

```
;   @(#)named.rev       4.1     (ULTRIX)        1/18/88
;
; Data file for 22.111.in-addr.arpa domain (inverse mapping).
;name     ttl    addr-class      entry-type      origin          person
@                IN              SOA             host.domain.    sysmgr.host.domain. (
                                                 91              ; Serial
                                                 43200           ; Refresh
                                                 3600            ; Retry
                                                 1209600         ; Expire
                                                 86400 )         ; Minimum
                 IN              NS              host.domain.
44.33            IN              PTR             hosttwo.domain.
```

A.6 The named_dump.db File

If you cause the named daemon to dump the data base, the results are stored in the /var/tmp/named_dump.db file. This file is helpful in checking the BIND data files for possible errors. Here is an excerpt of a named_dump.db file for a system whose data base is correctly set up:

```
; Dumped at Thu Jun 23 14:33:15 1988
; --- Cache & Data ---
$ORIGIN .
arpa 42391      IN   SOA SRI-NIC.ARPA. HOSTMASTER.SRI-NIC.ARPA. (
          880620 1800 300 604800 86400 )
     474256     IN   NS  BRL-AOS.ARPA.
     474256     IN   NS  A.ISI.EDU.
```

```
        474256    IN    NS    C.NYSER.NET.
        474256    IN    NS    TERP.UMD.EDU.
        474256    IN    NS    NS.NASA.GOV.
        42256     IN    SOA   SRI-NIC.ARPA. HOSTMASTER.SRI-NIC.ARPA. (
              880620 1800 300 604800 86400 )
$ORIGIN arpa.
GUNTER-ADAM    474256    IN    A     26.1.0.13  ; 15034
BRL-AOS 474256    IN    A     128.20.1.2; 13398
        474256    IN    A     192.5.25.82      ; 15097
SRI-NIC  474391    IN    A     26.0.0.73  ; 12557
        474391    IN    A     10.0.0.51  ; 13135
        42391     IN    MX    10 SRI-NIC.ARPA.
$ORIGIN 128.in-addr.arpa.
45    475231    IN    NS    SONORA.DEC.COM.
        475231    IN    NS    CYBELE.DEC.COM.
        475231    IN    NS    DECWRL.DEC.COM.
$ORIGIN 45.128.in-addr.arpa.
31    43231     IN    NS    know.roads.dec.com.
45          IN    SOA chicago.cities.dec.com.  doe.chicago.cities.dec.com. (
            7 1800 3600 1209600 86400 )
            IN    NS    chicago.cities.dec.com.
$ORIGIN 41.45.128.in-addr.arpa.
72          IN    PTR hole.cities.dec.com.
$ORIGIN 42.45.128.in-addr.arpa.
141         IN    PTR miami.cities.dec.com.
27          IN    PTR toledo.cities.dec.com.
8           IN    PTR paris.cities.dec.com.
1           IN    PTR potsdam.cities.dec.com.
$ORIGIN 43.45.128.in-addr.arpa.
2           IN    PTR madrid.cities.dec.com.
141         IN    PTR cannes.cities.dec.com.
$ORIGIN 44.45.128.in-addr.arpa.
24          IN    PTR galway.cities.dec.com.
27          IN    PTR antrim.cities.dec.com.
25          IN    PTR chism.cities.dec.com.
$ORIGIN 49.45.128.in-addr.arpa.
6           IN    PTR akron.cities.dec.com.
13          IN    PTR toledo.cities.dec.com.
14          IN    PTR madrid.cities.dec.com.
15          IN    PTR columbia.cities.dec.com.
2           IN    PTR stow.cities.dec.com.
4           IN    PTR atlanta.cities.dec.com.
$ORIGIN 0.127.in-addr.arpa.
0           IN    SOA chicago.cities.dec.com.  doe.chicago.cities.dec.com. (
```

```
        7 1800 3600 1209600 86400 )
        IN   NS   chicago.cities.dec.com.
$ORIGIN 0.0.127.in-addr.arpa.
1       IN   PTR localhost.
$ORIGIN com.
dec     IN   SOA decwrl.dec.com. postmaster.decwrl.dec.com. (
        142 43200 3600 1209600 86400 )
        IN   NS   sonora.dec.com.
        IN   NS   decwrl.dec.com.
        IN   NS   cybele.dec.com.
        IN   MX   100 decwrl.dec.com.
$ORIGIN dec.com.
aa      IN   A    128.45.1.87
bb      IN   A    128.45.1.81
cc   IN  CNAME  cc32.dec.com.
dd      IN   A    128.45.1.33
cities  IN   SOA chicago.cities.dec.com. doe.chicago.cities.dec.com. (
        7 1800 3600 1209600 86400 )
        IN   NS   chicago.cities.dec.com.
        IN   MX   200 detroit.dec.com.
tempe       IN   A    128.45.45.79
coxland     IN   A    128.45.1.176
$ORIGIN cities.dec.com.
ff      IN   A    128.45.45.221
*       IN   MX   51 tempe.cities.dec.com.
        IN   MX   50 chicago.cities.dec.com.
tempe       IN   A    128.45.45.79
        IN   TXT  "This is class IN data for tempe."
        3    TXT  "choas text data for tempe."
dixie   IN   CNAME  sunup.cities.dec.com.
        IN   CNAME  sunset.cities.dec.com.
alias   IN   SOA chicago.cities.dec.com. doe.chicago.cities.dec.com. (
        8 1800 3600 1209600 86400 )
        8 1800 3600 1209600 86400 )
am      IN   CNAME  antrim.cities.dec.com.
rutland     IN   A    128.45.45.105
derry   IN   CNAME  derry.cities.dec.com.
london      IN   CNAME  london.cities.dec.com.
$ORIGIN alias.cities.dec.com.
$ORIGIN uid.cities.dec.com.
$ORIGIN passwd.cities.dec.com.
$ORIGIN pa.dec.com.
wilton      IN   A    128.45.1.14
$ORIGIN nac.dec.com.
```

```
midland          IN   A     128.45.31.151
$ORIGIN NASA.GOV.
NS  474256   IN   A     128.102.16.10   ; 13964
$ORIGIN local.
tempe            IN   CNAME   tempe.cities.dec.com.
chicago          IN   CNAME   chicago.cities.dec.com.
$ORIGIN NYSER.NET.
C    474256   IN   A     192.33.4.12     ; 12999
$ORIGIN UMD.EDU.
TERP     474256   IN   A     10.1.0.17  ; 12186
     474256   IN   A     128.8.10.90     ; 3414
$ORIGIN ISI.EDU.
A    474256   IN   A     26.3.0.103; 15203
; --- Hints ---
$ORIGIN .
     474256   IN   NS    BRL-AOS.ARPA.
     474256   IN   NS    A.ISI.EDU.
     474256   IN   NS    GUNTER-ADAM.ARPA.
     474256   IN   NS    C.NYSER.NET.
     474256   IN   NS    TERP.UMD.EDU.
     474256   IN   NS    NS.NASA.GOV.
     3600 IN   SOA SRI-NIC.ARPA. HOSTMASTER.SRI-NIC.ARPA. (
          880620 1800 300 604800 86400 )
UK   407690   IN   NS    NS1.CS.UCL.AC.UK.
     407690   IN   NS    NS2.CS.UCL.AC.UK.
     407690   IN   NS    BRL-AOS.ARPA.
$ORIGIN arpa.
GUNTER-ADAM     474256    IN   A     26.1.0.13
BRL-AOS 474256    IN   A     128.20.1.2
     474256   IN   A     192.5.25.82
SRI-NIC  474256   IN   A     26.0.0.73  ; 2850
     474256   IN   A     10.0.0.51
     13620    IN   MX  10 SRI-NIC.ARPA.
$ORIGIN 128.in-addr.arpa.
45   348719   IN   NS    SONORA.DEC.COM.
     348719   IN   NS    CYBELE.DEC.COM.
     348719   IN   NS    DECWRL.DEC.COM.
$ORIGIN 45.128.in-addr.arpa.
31   3600 IN   NS    iknow.nac.dec.com.
$ORIGIN CS.UCL.AC.UK.
NS2 407690   IN   A     128.16.8.3
NS1 407690   IN   A     128.16.5.32
nss  249294   IN   A     128.41.9.3
     249294   IN   A     14.0.0.9
```

```
    249294    IN   MX  13 nss.cs.ucl.ac.uk.
    249294    IN   HINFO   "MICROVAX2" "ULTRIX1.2"
    249294    IN   WKS     128.41.9.3 tcp  telnet smtp
$ORIGIN NASA.GOV.
NS  474256    IN   A    128.102.16.10
$ORIGIN UMD.EDU.
TERP    474256    IN   A    10.1.0.17
    474256    IN   A    128.8.10.90
$ORIGIN ISI.EDU.
A   474256    IN   A    26.3.0.103
$ORIGIN NYSER.NET.
C   474256    IN   A    192.33.4.12
```

A.7 The named.run File

If you turn on debugging for the named daemon, the results are recorded in the /var/tmp/named.run file. This file is helpful in troubleshooting the BIND service. This section lists two sample named.run files. The first sample is indicative of a system that has the BIND service properly set up, while the second sample indicates a system that has the BIND service improperly set up.

A.7.1 A Healthy named.run File

The following sample named.run file logs the successful BIND service transactions. Notice the numerous ANSWERS.

```
Debug turned ON, Level 5
bootfile = /etc/named.primary/named.boot
ns_init( /etc/named.primary/named.boot)
savehash GROWING to 2
savehash GROWING to 2
zone[1] type 1: 'cities.dec.com', source = cities.dec.com.SOA
db_load( cities.dec.com.SOA, cities.dec.com, 1)
d='cities.dec.com', c=1, t=6, ttl=0, data='boston.cities.dec.com.'
db_update( cities.dec.com, 0x31c04, 0x31c04, 01, 0x203a4)
savehash GROWING to 2
savehash GROWING to 2
db_update: adding 31c04
d='cities.dec.com', c=1, t=2, ttl=0, data='boston.cities.dec.com.'
db_update( cities.dec.com, 0x32404, 0x32404, 01, 0x203a4)
match( 0x31c04, 1, 2) 1, 6
db_update: adding 32404
d='boston.cities.dec.com', c=1, t=1, ttl=0, data='128.45.45.93'
db_update( boston.cities.dec.com, 0x2d4c4, 0x2d4c4, 01, 0x203a4)
```

```
savehash GROWING to 2
db_update: adding 2d4c4
d='tampa.cities.dec.com', c=1, t=1, ttl=0, data='128.45.45.79'
db_update(tampa.cities.dec.com, 0x2d524, 0x2d524, 01, 0x203a4)
db_update: adding 2d524
d='*.cities.dec.com', c=1, t=15, ttl=0, data='51'
db_update(*.cities.dec.com, 0x324c4, 0x324c4, 01, 0x203a4)
db_update: adding 324c4
d='*.cities.dec.com', c=1, t=15, ttl=0, data='50'
db_update(*.cities.dec.com, 0x32504, 0x32504, 01, 0x203a4)
match(0x324c4, 1, 15) 1, 15
db_update: flags = 0x1, sizes = 20, 19 (1)
db_update: adding 32504
db_load(cities.dec.com.db, cities.dec.com, 1)
d='localhost.cities.dec.com', c=1, t=1, ttl=0, data='127.0.0.1'
db_update(localhost.cities.dec.com, 0x2d5a4, 0x2d5a4, 01, 0x203a4)
db_update: adding 2d5a4
d='nashua.cities.dec.com', c=1, t=1, ttl=0, data='128.45.45.17'
db_update(nashua.cities.dec.com, 0x2d5e4, 0x2d5e4, 01, 0x203a4)
savehash GROWING to 11
savehash(0x2d4e4) cnt=5, sz=2, newsz=11
db_update: adding 2d5e4
d='paris.cities.dec.com', c=1, t=1, ttl=0, data='128.45.42.1'
db_update(paris.cities.dec.com, 0x2d4e4, 0x2d4e4, 01, 0x203a4)
db_update: adding 2d4e4
d='paris.cities.dec.com', c=1, t=13, ttl=0, data='VAX'
db_update(paris.cities.dec.com, 0x32644, 0x32644, 01, 0x203a4)
match(0x2d4e4, 1, 13) 1, 1
db_update: adding 32644
d='p.cities.dec.com', c=1, t=5, ttl=0, data='paris'
db_update(p.cities.dec.com, 0x32684, 0x32684, 01, 0x203a4)
db_update: adding 32684
d='galway.cities.dec.com', c=1, t=1, ttl=0, data='128.45.45.1'
db_update(galway.cities.dec.com, 0x2d664, 0x2d664, 01, 0x203a4)
db_update: adding 2d664
d='gy.cities.dec.com', c=1, t=5, ttl=0, data='galway'
db_update(gy.cities.dec.com, 0x32704, 0x32704, 01, 0x203a4)
db_update: adding 32704
d='norfolk.cities.dec.com', c=1, t=1, ttl=0, data='128.45.42.2'
db_update(norfolk.cities.dec.com, 0x2d6c4, 0x2d6c4, 01, 0x203a4)
db_update: adding 2d6c4
d='n.cities.dec.com', c=1, t=5, ttl=0, data='norfolk'
db_update(n.cities.dec.com, 0x32784, 0x32784, 01, 0x203a4)
db_update: adding 32784
```

```
d='bangor.cities.dec.com', c=1, t=1, ttl=0, data='128.45.45.2'
db_update(bangor.cities.dec.com, 0x2d724, 0x2d724, 01, 0x203a4)
db_update: adding 2d724
d='bg.cities.dec.com', c=1, t=5, ttl=0, data='bangor'
db_update(bg.cities.dec.com, 0x32804, 0x32804, 01, 0x203a4)
db_update: adding 32804
d='canton.cities.dec.com', c=1, t=1, ttl=0, data='128.45.43.2'
db_update(canton.cities.dec.com, 0x2d784, 0x2d784, 01, 0x203a4)
db_update: adding 2d784
d='c.cities.dec.com', c=1, t=5, ttl=0, data='canton'
db_update(c.cities.dec.com, 0x32884, 0x32884, 01, 0x203a4)
db_update: adding 32884
d='few.cities.dec.com', c=1, t=1, ttl=0, data='128.45.45.3'
db_update(few.cities.dec.com, 0x2d7e4, 0x2d7e4, 01, 0x203a4)
db_update: adding 2d7e4
d='f.cities.dec.com', c=1, t=5, ttl=0, data='few'
db_update(f.cities.dec.com, 0x32904, 0x32904, 01, 0x203a4)
db_update: adding 32904
d='trouble.cities.dec.com', c=1, t=1, ttl=0, data='128.45.45.4'
db_update(trouble.cities.dec.com, 0x2d844, 0x2d844, 01, 0x203a4)
db_update: adding 2d844
d='t.cities.dec.com', c=1, t=5, ttl=0, data='trouble'
db_update(t.cities.dec.com, 0x32984, 0x32984, 01, 0x203a4)
db_update: adding 32984
d='foto.cities.dec.com', c=1, t=1, ttl=0, data='128.45.45.5'
db_update(foto.cities.dec.com, 0x2d8a4, 0x2d8a4, 01, 0x203a4)
db_update: adding 2d8a4
d='wise.cities.dec.com', c=1, t=1, ttl=0, data='128.45.45.6'
db_update(wise.cities.dec.com, 0x2d8e4, 0x2d8e4, 01, 0x203a4)
db_update: adding 2d8e4
d='w.cities.dec.com', c=1, t=5, ttl=0, data='wise'
db_update(w.cities.dec.com, 0x32a44, 0x32a44, 01, 0x203a4)
db_update: adding 32a44
d='marg.cities.dec.com', c=1, t=1, ttl=0, data='128.45.45.7'
db_update(marg.cities.dec.com, 0x2d944, 0x2d944, 01, 0x203a4)
savehash GROWING to 113
savehash(0x32584) cnt=23, sz=11, newsz=113
db_load: origin NYSER.NET., buf 45.128.in-addr.arpa
db_load: origin now NYSER.NET
d='C.NYSER.NET', c=1, t=1, ttl=583091719, data='192.33.4.12'
db_update(C.NYSER.NET, 0x5dba4, 0x5dba4, 021, 0x203c4) hint
savehash GROWING to 2
db_update: adding hint 5dba4
db_load: origin UMD.EDU., buf NYSER.NET
```

```
db_load: origin now UMD.EDU
d='TERP.UMD.EDU', c=1, t=1, ttl=583091719, data='10.1.0.17'
db_update(TERP.UMD.EDU, 0x5dc64, 0x5dc64, 021, 0x203c4) hint
savehash GROWING to 2
db_update: adding hint 5dc64
d='TERP.UMD.EDU', c=1, t=1, ttl=583091719, data='128.8.10.90'
db_update(TERP.UMD.EDU, 0x5dd24, 0x5dd24, 021, 0x203c4) hint
match(0x5dc64, 1, 1) 1, 1
db_update: flags = 0x11, sizes = 4, 4 (4)
db_update: adding hint 5dd24
db_load: origin ISI.EDU., buf UMD.EDU
db_load: origin now ISI.EDU
d='A.ISI.EDU', c=1, t=1, ttl=583091719, data='26.3.0.103'
db_update(A.ISI.EDU, 0x5dd44, 0x5dd44, 021, 0x203c4) hint
savehash GROWING to 2
db_update: adding hint 5dd44
z_time 582916333, z_refresh 3600
exit ns_init() Next interrupt in 3598 sec
database initialized
net x2d2d80 mask xffffff my_addr x5d2d2d80 128.45.45.93
net x7f mask xff my_addr x100007f 127.0.0.1
net x2d80 mask xffff my_addr x5d2d2d80 128.45.45.93
ds 0.0.0.0 7
dqp->dq_addr 127.0.0.1 d_dfd 8
dqp->dq_addr 128.45.45.93 d_dfd 9
Ready to answer queries.
prime_cache: priming = 0
sysquery(, 1, 2)
findns: using hints
findns: np 0x5d964
findns: 7 NS's added for ''
qnew(x1f604)
nslookup(nsp=x7fffdcbc,qp=x1f604)
nslookup: NS SRI-NIC.ARPA c1 t2 (x1)
nslookup: 2 ns addrs
nslookup: NS BRL-AOS.ARPA c1 t2 (x1)
nslookup: 4 ns addrs
nslookup: NS A.ISI.EDU c1 t2 (x1)
nslookup: 5 ns addrs
nslookup: NS GUNTER-ADAM.ARPA c1 t2 (x1)
nslookup: 6 ns addrs
nslookup: NS C.NYSER.NET c1 t2 (x1)
nslookup: 7 ns addrs
nslookup: NS TERP.UMD.EDU c1 t2 (x1)
```

```
nslookup: 9 ns addrs
nslookup: NS NS.NASA.GOV c1 t2 (x1)
NS.NASA.GOV: not found ??? 0
nslookup: 9 ns addrs total
schedretry(0x1f604, 13sec)
sysquery: send -> 26.0.0.73 7 (53), nsid=1 id=0 0ms
Return from getdtablesize() > FD_SETSIZE
datagram from 128.45.45.15, 9  1568 (39)
ns_req()
HEADER:
    opcode = QUERY, id = 155, rcode = NOERROR
    header flags:  rd
    qdcount = 1, ancount = 0, nscount = 0, arcount = 0

QUESTIONS:
    localhost.cities.dec.com, type = A, class = IN

req: nlookup(localhost.cities.dec.com) id 39680 type=1
req: found 'localhost.cities.dec.com' as 'localhost.cities.dec.com' (cname=0)
wanted(2d5a4, 1, 1) 1, 1
make_rr(localhost.cities.dec.com, 2d5a4, 7fffddbb, 473, 1) 4
finddata: added 1 class 1 type 1 RRs
req: foundname = 1 count = 1 founddata = 1 cname = 0
sort_response(1)
findns: np 0x2d5c4
match(0x2d5a4, 1, 6) 1, 1
findns: np 0x2d4a4
match(0x31c04, 1, 6) 1, 6
findns: SOA found
req: leaving (localhost.cities.dec.com, rcode 0)
req: answer -> 128.45.45.15 9 (1568) id=155 Local
datagram from 128.45.45.15, 9  1570 (40)
ns_req()
HEADER:
    opcode = QUERY, id = 154, rcode = NOERROR
    header flags:  rd
    qdcount = 1, ancount = 0, nscount = 0, arcount = 0

QUESTIONS:
    1.0.0.127.in-addr.arpa, type = PTR, class = IN

req: nlookup(1.0.0.127.in-addr.arpa) id 39424 type=12
req: found '1.0.0.127.in-addr.arpa' as '1.0.0.127.in-addr.arpa' (cname=0)
wanted(44c44, 1, 12) 1, 12
```

```
make_rr( 1.0.0.127.in-addr.arpa, 44c44, 7fffddbc, 472, 1) 10
finddata: added 1 class 1 type 12 RRs
req: foundname = 1 count = 1 founddata = 1 cname = 0
sort_response( 1)
findns: np 0x44a44
match( 0x44c44, 1, 6) 1, 12
findns: np 0x44a04
match( 0x32084, 1, 6) 1, 6
findns: SOA found
req: leaving ( 1.0.0.127.in-addr.arpa, rcode 0)
req: answer -> 128.45.45.15 9 (1570) id=154 Local
retry( x1f604) id=0
resend( addr=1 n=0) -> 10.0.0.51 7 (53) nsid=1 id=0 0ms
unsched( 0x1f604, 0 )
schedretry( 0x1f604, 13sec)
datagram from 10.0.0.51, 9  53 (421)
ns_req( )
HEADER:
    opcode = QUERY, id = 1, rcode = NOERROR
    header flags:  qr aa
    qdcount = 1, ancount = 7, nscount = 1, arcount = 10

QUESTIONS:
    ., type = NS, class = IN

ANSWERS:
    .
    type = NS, class = IN, ttl = 518400, dlen = 14
    domain name = SRI-NIC.ARPA

    .
    type = NS, class = IN, ttl = 518400, dlen = 14
    domain name = BRL-AOS.ARPA

    .
    type = NS, class = IN, ttl = 518400, dlen = 11
    domain name = A.ISI.EDU

    .
    type = NS, class = IN, ttl = 518400, dlen = 18
    domain name = GUNTER-ADAM.ARPA

    .
    type = NS, class = IN, ttl = 518400, dlen = 13
```

domain name = C.NYSER.NET

.
type = NS, class = IN, ttl = 518400, dlen = 14
domain name = TERP.UMD.EDU

.
type = NS, class = IN, ttl = 518400, dlen = 13
domain name = NS.NASA.GOV

NAME SERVERS:

.
type = SOA, class = IN, ttl = 86400, dlen = 59
origin = SRI-NIC.ARPA
mail addr = HOSTMASTER.SRI-NIC.ARPA
serial=880620, refresh=1800, retry=300, expire=604800, min=86400

ADDITIONAL RECORDS:

SRI-NIC.ARPA
type = A, class = IN, ttl = 518400, dlen = 4
internet address = 26.0.0.73

SRI-NIC.ARPA
type = A, class = IN, ttl = 518400, dlen = 4
internet address = 10.0.0.51

BRL-AOS.ARPA
type = A, class = IN, ttl = 518400, dlen = 4
internet address = 128.20.1.2

BRL-AOS.ARPA
type = A, class = IN, ttl = 518400, dlen = 4
internet address = 192.5.25.82

A.ISI.EDU
type = A, class = IN, ttl = 518400, dlen = 4
internet address = 26.3.0.103

GUNTER-ADAM.ARPA
type = A, class = IN, ttl = 518400, dlen = 4
internet address = 26.1.0.13

C.NYSER.NET
type = A, class = IN, ttl = 518400, dlen = 4

```
        internet address = 192.33.4.12

        TERP.UMD.EDU
        type = A, class = IN, ttl = 518400, dlen = 4
        internet address = 10.1.0.17

        TERP.UMD.EDU
        type = A, class = IN, ttl = 518400, dlen = 4
        internet address = 128.8.10.90

        NS.NASA.GOV
        type = A, class = IN, ttl = 518400, dlen = 4
        internet address = 128.102.16.10

qfindid(1)
SYSTEM response nsid=1 id=0
stime 582912747/180000  now 582912750/390000 rtt 3210
NS #1 addr 10.0.0.51 used, rtt 3210
NS #0 26.0.0.73 rtt now 3852
NS #2 128.20.1.2 rtt now 0
NS #3 192.5.25.82 rtt now 0
NS #4 26.3.0.103 rtt now 0
NS #5 26.1.0.13 rtt now 0
NS #6 192.33.4.12 rtt now 0
NS #7 10.1.0.17 rtt now 0
NS #8 128.8.10.90 rtt now 0
resp: ancount 7, nscount 1, arcount 10
sort_response(7)
sort_rr( x7fffdda5, 7, 0.0.0.0)
sort_rr( x7fffdda5, 7, 128.45.45.0)
sort_rr( x7fffdda5, 7, 127.0.0.0)
sort_rr( x7fffdda5, 7, 128.45.0.0)
doupdate(zone 0, savens 7fffd3a0, flags 9)
doupdate: dname  type 2 class 1 ttl 518400
db_update(, 0x5c1c4, 0x5c1c4, 011, 0x203a4)
db_update: hint '' 583431150
db_update(, 0x5c204, 0x5c204, 031, 0x203c4) hint
match(0x5bf04, 1, 2) 1, 2
db_update: flags = 0x19, sizes = 13, 13 (0)
db_update: new ttl 583431150, +518400
db_update: hint 5c204 freed
db_update: adding 5c1c4
doupdate(zone 0, savens 7fffd3a0, flags 9)
doupdate: dname  type 2 class 1 ttl 518400
```

```
db_update(, 0x5c204, 0x5c204, 011, 0x203a4)
db_update: hint '' 583431150
db_update(, 0x5c244, 0x5c244, 031, 0x203c4) hint
match(0x5bf04, 1, 2) 1, 2
db_update: flags = 0x19, sizes = 13, 13 (17)
match(0x5bf44, 1, 2) 1, 2
db_update: flags = 0x19, sizes = 13, 13 (0)
db_update: new ttl 583431150, +518400
db_update: hint 5c244 freed
match(0x5c1c4, 1, 2) 1, 2
db_update: flags = 0x9, sizes = 13, 13 (17)
db_update: adding 5c204
doupdate(zone 0, savens 7fffd3a0, flags 9)
doupdate: dname  type 2 class 1 ttl 518400
db_update(, 0x5c244, 0x5c244, 011, 0x203a4)
db_update: hint '' 583431150
HEADER:
    opcode = QUERY, id = 220, rcode = NOERROR
    header flags:  rd
    qdcount = 1, ancount = 0, nscount = 0, arcount = 0

QUESTIONS:
    1.0.0.127.in-addr.arpa, type = PTR, class = IN

req: nlookup(1.0.0.127.in-addr.arpa) id 56320 type=12
req: found '1.0.0.127.in-addr.arpa' as '1.0.0.127.in-addr.arpa' (cname=0)
wanted(44c44, 1, 12) 1, 12
make_rr(1.0.0.127.in-addr.arpa, 44c44, 7fffddbc, 472, 1) 10
finddata: added 1 class 1 type 12 RRs
req: foundname = 1 count = 1 founddata = 1 cname = 0
sort_response(1)
findns: np 0x44a44
match(0x44c44, 1, 6) 1, 12
findns: np 0x44a04
match(0x32084, 1, 6) 1, 6
findns: SOA found
req: leaving (1.0.0.127.in-addr.arpa, rcode 0)
req: answer -> 128.45.45.95 9 (4713) id=220 Local
datagram from 128.45.45.95, 9  4714 (40)
ns_req()
HEADER:
    opcode = QUERY, id = 3, rcode = NOERROR
    header flags:  rd
    qdcount = 1, ancount = 0, nscount = 0, arcount = 0
```

```
QUESTIONS:
     antrim.cities.dec.com.com, type = A, class = IN

req: nlookup(antrim.cities.dec.com.com) id 768 type=1
req: found 'antrim.cities.dec.com.com' as 'com' (cname=0)
findns: np 0x2d424
findns: using cache
findns: np 0x5de24
findns: 7 NS's added for ''
ns_forw()
qnew(x1f604)
nslookup(nsp=x7fffdb80,qp=x1f604)
nslookup: NS SRI-NIC.ARPA c1 t2 (x0)
nslookup: 2 ns addrs
nslookup: NS BRL-AOS.ARPA c1 t2 (x0)
nslookup: 4 ns addrs
nslookup: NS A.ISI.EDU c1 t2 (x0)
nslookup: 5 ns addrs
nslookup: NS GUNTER-ADAM.ARPA c1 t2 (x0)
nslookup: 6 ns addrs
nslookup: NS C.NYSER.NET c1 t2 (x0)
nslookup: 7 ns addrs
nslookup: NS TERP.UMD.EDU c1 t2 (x0)
nslookup: 9 ns addrs
nslookup: NS NS.NASA.GOV c1 t2 (x0)
nslookup: 10 ns addrs total
schedretry(0x1f604, 4sec)
forw: forw -> 192.33.4.12 7 (53) nsid=4 id=3 0ms
datagram from 128.45.45.95, 9  4715 (40)
ns_req()
HEADER:
     opcode = QUERY, id = 1023, rcode = NOERROR
     header flags:  rd
     qdcount = 1, ancount = 0, nscount = 0, arcount = 0

QUESTIONS:
     queens.dec.com, type = A, class = IN

req: nlookup(queens.dec.com) id 65283 type=1
req: found 'queens.dec.com' as 'queens.dec.com' (cname=0)
wanted(4c1c4, 1, 1) 1, 15
finddata: added 0 class 1 type 1 RRs
findns: np 0x4c7a4
match(0x4c1c4, 1, 6) 1, 15
```

```
findns: np 0x2d464
match(0x32104, 1, 6) 1, 6
findns: SOA found
req: leaving (queens.dec.com, rcode 0)
make_rr(dec.com, 32104, 7fffddb3, 481, 1) 61
req: answer -> 128.45.45.87 9 (1741) id=1023 Local
datagram from 128.45.45.87, 9  1743 (31)
ns_req()
HEADER:
    opcode = QUERY, id = 1023, rcode = NOERROR
    header flags:  rd
    qdcount = 1, ancount = 0, nscount = 0, arcount = 0

QUESTIONS:
    queens.dec.com, type = A, class = IN

req: nlookup(queens.dec.com) id 65283 type=1
req: found 'queens.dec.com' as 'queens.dec.com' (cname=0)
wanted(4c1c4, 1, 1) 1, 15
finddata: added 0 class 1 type 1 RRs
findns: np 0x4c7a4
match(0x4c1c4, 1, 6) 1, 15
findns: np 0x2d464
match(0x32104, 1, 6) 1, 6
findns: SOA found
req: leaving (queens.dec.com, rcode 0)
make_rr(dec.com, 32104, 7fffddb3, 481, 1) 61
req: answer -> 128.45.45.87 9 (1743) id=1023 Local
datagram from 128.45.45.87, 9  1745 (31)
ns_req()
HEADER:
    opcode = QUERY, id = 1024, rcode = NOERROR
    header flags:  rd
    qdcount = 1, ancount = 0, nscount = 0, arcount = 0

QUESTIONS:
    queens.dec.com, type = A, class = IN

req: nlookup(queens.dec.com) id 4 type=1
req: found 'queens.dec.com' as 'queens.dec.com' (cname=0)
wanted(4c1c4, 1, 1) 1, 15
finddata: added 0 class 1 type 1 RRs
findns: np 0x4c7a4
match(0x4c1c4, 1, 6) 1, 15
```

```
findns: np 0x2d464
match(0x32104, 1, 6) 1, 6
findns: SOA found
req: leaving (queens.dec.com, rcode 0)
make_rr(dec.com, 32104, 7fffddb3, 481, 1) 61
req: answer -> 128.45.45.87 9 (1745) id=1024 Local
datagram from 128.45.45.87, 9  1747 (31)
ns_req()
HEADER:
    opcode = QUERY, id = 1024, rcode = NOERROR
    header flags:  rd
    qdcount = 1, ancount = 0, nscount = 0, arcount = 0

QUESTIONS:
    queens.dec.com, type = A, class = IN

req: nlookup(queens.dec.com) id 4 type=1
req: found 'queens.dec.com' as 'queens.dec.com' (cname=0)
wanted(4c1c4, 1, 1) 1, 15
finddata: added 0 class 1 type 1 RRs
findns: np 0x4c7a4
match(0x4c1c4, 1, 6) 1, 15
findns: np 0x2d464
match(0x32104, 1, 6) 1, 6
findns: SOA found
req: leaving (queens.dec.com, rcode 0)
make_rr(dec.com, 32104, 7fffddb3, 481, 1) 61
req: answer -> 128.45.45.87 9 (1747) id=1024 Local
datagram from 128.45.45.87, 9  1749 (31)
ns_req()
HEADER:
    opcode = QUERY, id = 1025, rcode = NOERROR
    header flags:  rd
    qdcount = 1, ancount = 0, nscount = 0, arcount = 0

QUESTIONS:
    queens.dec.com, type = A, class = IN

req: nlookup(queens.dec.com) id 260 type=1
req: found 'queens.dec.com' as 'queens.dec.com' (cname=0)
wanted(4c1c4, 1, 1) 1, 15
finddata: added 0 class 1 type 1 RRs
findns: np 0x4c7a4
match(0x4c1c4, 1, 6) 1, 15
```

```
findns: np 0x2d464
match(0x32104, 1, 6) 1, 6
findns: SOA found
req: leaving (queens.dec.com, rcode 0)
make_rr(dec.com, 32104, 7fffddb3, 481, 1) 61
req: answer -> 128.45.45.87 9 (1749) id=1025 Local
datagram from 128.45.45.87, 9  1751 (31)
ns_req()
HEADER:
    opcode = QUERY, id = 1025, rcode = NOERROR
    header flags:  rd
    qdcount = 1, ancount = 0, nscount = 0, arcount = 0

QUESTIONS:
    queens.dec.com, type = A, class = IN

req: nlookup(queens.dec.com) id 260 type=1
req: found 'queens.dec.com' as 'queens.dec.com' (cname=0)
wanted(4c1c4, 1, 1) 1, 15
finddata: added 0 class 1 type 1 RRs
findns: np 0x4c7a4
match(0x4c1c4, 1, 6) 1, 15
findns: np 0x2d464
match(0x32104, 1, 6) 1, 6
findns: SOA found
req: leaving (queens.dec.com, rcode 0)
make_rr(dec.com, 32104, 7fffddb3, 481, 1) 61
req: answer -> 128.45.45.87 9 (1751) id=1025 Local
datagram from 128.45.45.15, 9  1574 (39)
ns_req()
HEADER:
    opcode = QUERY, id = 157, rcode = NOERROR
    header flags:  rd
    qdcount = 1, ancount = 0, nscount = 0, arcount = 0

QUESTIONS:
    localhost.cities.dec.com, type = A, class = IN

req: nlookup(localhost.cities.dec.com) id 40192 type=1
req: found 'localhost.cities.dec.com' as 'localhost.cities.dec.com' (cname=0)
wanted(2d5a4, 1, 1) 1, 1
make_rr(localhost.cities.dec.com, 2d5a4, 7fffddbb, 473, 1) 4
finddata: added 1 class 1 type 1 RRs
req: foundname = 1 count = 1 founddata = 1 cname = 0
```

```
sort_response( 1)
findns: np 0x2d5c4
match( 0x2d5a4, 1, 6) 1, 1
findns: np 0x2d4a4
match( 0x31c04, 1, 6) 1, 6
findns: SOA found
req: leaving (localhost.cities.dec.com, rcode 0)
req: answer -> 128.45.45.15 9 (1574) id=157 Local
datagram from 128.45.45.15, 9  1576 (40)
ns_req( )
HEADER:
     opcode = QUERY, id = 156, rcode = NOERROR
     header flags:  rd
     qdcount = 1, ancount = 0, nscount = 0, arcount = 0

QUESTIONS:
     1.0.0.127.in-addr.arpa, type = PTR, class = IN

req: nlookup( 1.0.0.127.in-addr.arpa) id 39936 type=12
req: found '1.0.0.127.in-addr.arpa' as '1.0.0.127.in-addr.arpa' (cname=0)
wanted( 44c44, 1, 12) 1, 12
make_rr( 1.0.0.127.in-addr.arpa, 44c44, 7fffddbc, 472, 1) 10
finddata: added 1 class 1 type 12 RRs
req: foundname = 1 count = 1 founddata = 1 cname = 0
sort_response( 1)
findns: np 0x44a44
match( 0x44c44, 1, 6) 1, 12
findns: np 0x44a04
match( 0x32084, 1, 6) 1, 6
findns: SOA found
req: leaving (1.0.0.127.in-addr.arpa, rcode 0)
Req: answer -> 128.45.45.15 9 (1576) id=156 Local
Debug turned OFF, Level 5
```

A.7.2 An Unhealthy named.run File

The following sample named.run file indicates that there is an error with the BIND service. Notice the numerous QUESTIONS that are not followed by ANSWERS. Notice, too, the numerous attempts to load BIND name servers to answer queries.

```
Debug turned ON, Level 1
Debug turned ON, Level 2
Debug turned ON, Level 3
Debug turned ON, Level 4
```

```
Debug turned ON, Level 5

datagram from 128.45.45.93 port 2034, fd 7, len 39
ns_req()
HEADER:
        opcode = QUERY, id = 1, rcode = NOERROR
        header flags:  rd
        qdcount = 1, ancount = 0, nscount = 0, arcount = 0

QUESTIONS:
        localhost.cities.dec.com, type = A, class = IN

req: nlookup(localhost.cities.dec.com) id 256 type=1
req: found 'localhost.cities.dec.com' as 'localhost.cities.dec.com' (cname=0)
wanted(2d564, 1, 1) 1, 1
make_rr(localhost.cities.dec.com, 2d564, 7fffddbb, 473, 1) 4 zone 1 ttl 0
finddata: added 1 class 1 type 1 RRs
req: foundname = 1 count = 1 founddata = 1 cname = 0
sort_response(1)
findns: np 0x2d584
match(0x2d564, 1, 6) 1, 1
findns: np 0x2d4a4
match(0x31c04, 1, 6) 1, 6
findns: SOA found
req: leaving (localhost.cities.dec.com, rcode 0)
req: answer -> 128.45.45.93 9 (2034) id=1 Local

datagram from 128.45.45.93 port 2036, fd 7, len 47
ns_req()
HEADER:
        opcode = QUERY, id = 2, rcode = NOERROR
        header flags:  rd
        qdcount = 1, ancount = 0, nscount = 0, arcount = 0

QUESTIONS:
        wepel.cities.dec.com.cities.dec.com, type = A, class = IN

req: nlookup(wepel.cities.dec.com.cities.dec.com) id 512 type=1
req: found 'wepel.cities.dec.com.cities.dec.com' as
  'wepel.cities.dec.com.cities.dec.com' (cname=0)
wanted(32444, 1, 1) 1, 15
finddata: added 0 class 1 type 1 RRs
findns: np 0x2d4e4
match(0x32444, 1, 6) 1, 15
```

```
match(0x31c04, 1, 6) 1, 6
findns: SOA found
req: leaving (wepel.cities.dec.com.cities.dec.com, rcode 0)
req: answer -> 128.45.45.93 9 (2036) id=2 Local

datagram from 128.45.45.93 port 2037, fd 7, len 43
ns_req()
HEADER:
        opcode = QUERY, id = 3, rcode = NOERROR
        header flags:  rd
        qdcount = 1, ancount = 0, nscount = 0, arcount = 0

QUESTIONS:
        wepel.cities.dec.com.dec.com, type = A, class = IN

req: nlookup(wepel.cities.dec.com.dec.com) id 768 type=1
req: found 'wepel.cities.dec.com.dec.com' as 'wepel.cities.dec.com.dec.com' (cname=0)
wanted(4bec4, 1, 1) 1, 15
finddata: added 0 class 1 type 1 RRs
findns: np 0x4c624
match(0x4bec4, 1, 6) 1, 15
findns: np 0x2d464
match(0x31d84, 1, 6) 1, 6
findns: SOA found
req: leaving (wepel.cities.dec.com.dec.com, rcode 0)
req: answer -> 128.45.45.93 9 (2037) id=3 Local

datagram from 128.45.45.93 port 2038, fd 7, len 35
ns_req()
HEADER:
        opcode = QUERY, id = 4, rcode = NOERROR
        header flags:  rd
        qdcount = 1, ancount = 0, nscount = 0, arcount = 0

QUESTIONS:
        wepel.cities.dec.com, type = A, class = IN

req: nlookup(wepel.cities.dec.com) id 1024 type=1
req: found 'wepel.cities.dec.com' as 'wepel.cities.dec.com' (cname=0)
wanted(2d504, 1, 1) 1, 1
make_rr(wepel.cities.dec.com, 2d504, 7fffddb7, 477, 1) 4 zone 1 ttl 0
finddata: added 1 class 1 type 1 RRs
req: foundname = 1 count = 1 founddata = 1 cname = 0
sort_response(1)
```

```
findns: np 0x2d524
match(0x2d504, 1, 6) 1, 1
findns: np 0x2d4a4
match(0x31c04, 1, 6) 1, 6
findns: SOA found
req: leaving (wepel.cities.dec.com, rcode 0)
req: answer -> 128.45.45.93 9 (2038) id=4 Local

datagram from 128.45.45.93 port 2039, fd 7, len 28
ns_req()
HEADER:
    opcode = QUERY, id = 5, rcode = NOERROR
    header flags:  rd
    qdcount = 1, ancount = 0, nscount = 0, arcount = 0

QUESTIONS:
    muon.local, type = ANY, class = IN

req: nlookup(muon.local) id 1280 type=255
req: missed 'muon.local' as '' (cname=0)
findns: using cache
findns: np 0x5dec4
findns: 7 NS's added for ''
ns_forw()
qnew(x1f604)
nslookup(nsp=x7fffdb88,qp=x1f604)
nslookup: NS SRI-NIC.ARPA c1 t2 (x0)
nslookup: 2 ns addrs
nslookup: NS BRL-AOS.ARPA c1 t2 (x0)
nslookup: 4 ns addrs
nslookup: NS A.ISI.EDU c1 t2 (x0)
nslookup: 5 ns addrs
nslookup: NS GUNTER-ADAM.ARPA c1 t2 (x0)
nslookup: 6 ns addrs
nslookup: NS C.NYSER.NET c1 t2 (x0)
nslookup: 8 ns addrs
nslookup: NS TERP.UMD.EDU c1 t2 (x0)
nslookup: 10 ns addrs total
retrytime: nstime 1ms.
schedretry(0x1f604, 4sec)
forw: forw -> 10.1.0.17 7 (53) nsid=9 id=5 1260ms retry 4 sec

datagram from 10.1.0.17 port 53, fd 7, len 86
ns_req()
```

```
HEADER:
    opcode = QUERY, id = 9, rcode = NXDOMAIN
    header flags:  qr aa ra
    qdcount = 1, ancount = 0, nscount = 1, arcount = 0

QUESTIONS:
    muon.local, type = ANY, class = IN

NAME SERVERS:
    .
    type = SOA, class = IN, ttl = 86400, dlen = 47
    origin = SRI-NIC.ARPA
    mail addr = HOSTMASTER.SRI-NIC.ARPA
    serial=880513, refresh=1800, retry=300, expire=604800, min=86400

qfindid(9)
USER response nsid=9 id=5
stime 579814526/240000  now 579814528/940000 rtt 2700
NS #0 addr 10.1.0.17 used, rtt 1692
NS #1 128.213.5.17 rtt now 1433
NS #2 26.1.0.13 rtt now 1481
NS #3 26.3.0.103 rtt now 3364
NS #4 192.33.4.12 rtt now 3881
NS #5 128.8.10.90 rtt now 3995
NS #6 128.20.1.2 rtt now 4328
NS #7 10.0.0.51 rtt now 5194
NS #8 192.5.25.82 rtt now 5194
NS #9 26.0.0.73 rtt now 5194
resp: ancount 0, aucount 1, arcount 0
doupdate(zone 0, savens 7fffd3b0, flags 19)
doupdate: dname  type 6 class 1 ttl 86400
db_update(, 0x32084, 0x32084, 031, 0x5da44)
match(0x5d744, 1, 6) 1, 2
match(0x5d784, 1, 6) 1, 2
match(0x5d7c4, 1, 6) 1, 2
match(0x5d804, 1, 6) 1, 2
match(0x5d844, 1, 6) 1, 2
match(0x5d884, 1, 6) 1, 2
match(0x5d8c4, 1, 6) 1, 2
match(0x32004, 1, 6) 1, 6
db_update: flags = 0x19, sizes = 57, 57 (0)
db_update: new ttl 579900928, +86400
update failed (DATAEXISTS)
resp: leaving auth NO
```

```
send_msg -> 128.45.45.93 (UDP 9 2039) id=5
qp 1f604 q_id: 1280  q_nsid: 2304 q_msglen: 28 q_naddr: 10 q_curaddr: 0
q_next: 0 q_link: 0
qremove(x1f604)
unsched(0x1f604, 5 )
qfree( x1f604 )

datagram from 128.45.45.93 port 2040, fd 7, len 34
ns_req()
HEADER:
     opcode = QUERY, id = 6, rcode = NOERROR
     header flags:  rd
     qdcount = 1, ancount = 0, nscount = 0, arcount = 0

QUESTIONS:
     tampa.cities.dec.com, type = MX, class = IN

req: nlookup(tampa.cities.dec.com) id 1536 type=15
req: found 'tampa.cities.dec.com' as 'muon.cities.dec.com' (cname=0)
wanted(3b4e4, 1, 15) 1, 1
finddata: added 0 class 1 type 15 RRs
findns: np 0x3b504
match(0x3b4e4, 1, 6) 1, 1
findns: np 0x2d4a4
match(0x31c04, 1, 6) 1, 6
findns: SOA found
req: leaving (tampa.cities.dec.com, rcode 0)
req: answer -> 128.45.45.93 9 (2040) id=6 Local

datagram from 128.45.45.93 port 2041, fd 7, len 34
ns_req()
HEADER:
     opcode = QUERY, id = 7, rcode = NOERROR
     header flags:  rd
     qdcount = 1, ancount = 0, nscount = 0, arcount = 0

QUESTIONS:
     tampa.cities.dec.com, type = A, class = IN

req: nlookup(tampa.cities.dec.com) id 1792 type=1
req: found 'tampa.cities.dec.com' as 'muon.cities.dec.com' (cname=0)
wanted(3b4e4, 1, 1) 1, 1
make_rr(tampa.cities.dec.com, 3b4e4, 7fffddb6, 478, 1) 4 zone 1 ttl 0
finddata: added 1 class 1 type 1 RRs
```

```
req: foundname = 1 count = 1 founddata = 1 cname = 0
sort_response(1)
findns: np 0x3b504
match(0x3b4e4, 1, 6) 1, 1
findns: np 0x2d4a4
match(0x31c04, 1, 6) 1, 6
findns: SOA found
req: leaving (tampa.cities.dec.com, rcode 0)
req: answer -> 128.45.45.93 9 (2041) id=7 Local
Debug turned OFF, Level 5
```

Appendix B

This appendix provides a copy of the BIND questionnaire that you need to complete and send to the NIC domain registrar to register your BIND domain. To obtain an on-line copy of the questionnaire, you can use the ftp command.

The following example shows a successful ftp exchange. In this example the site sri-nic.arpa is opened, the help option is invoked, and the BIND domain registration questionnaire is copied to the file /tmp/questionnaire on the local system:

```
# ftp
ftp> open
(to) sri-nic.arpa
Connected to sri-nic.arpa.
220 SRI-NIC.ARPA FTP Server Process 5Z(47)-6 at Fri 10-Jun-88 12:07-PDT
Name (sri-nic.arpa:liza): anonymous
Password (sri-nic.arpa:anonymous):
331 ANONYMOUS user ok, send real ident as password.
230 User ANONYMOUS logged in at Fri 10-Jun-88 12:07-PDT

ftp> help
Commands may be abbreviated.  Commands are:

!               dir             mget            quit            trace
append          form            mkdir           quote           type
ascii           get             mls             recv            user
bell            glob            mode            binary          hash
cd              lcd             prompt          send            ?

ftp> get
(remote-file) netinfo:domain-template.txt
(local-file) /tmp/questionnaire
200 Port 4.30 at host 128.45.45.93 accepted.
150 ASCII retrieve of <NETINFO>DOMAIN-TEMPLATE.TXT.28 started.
226 Transfer completed. 6129 (8) bytes transferred.
6129 bytes received in 4.62 seconds (1.3 Kbytes/s)
```

(continued on next page)

```
ftp> close
221 QUIT command received. Goodbye.
ftp> bye
```

Upon completing a successful ftp exchange, as shown in the previous example, here is what you receive:

```
# more /tmp/questionnaire
```

[NETINFO:DOMAIN-TEMPLATE.TXT] [2/88]

To establish a domain, the following information must be sent to the NIC Domain Registrar (HOSTMASTER@SRI-NIC.ARPA). Questions may be addressed to the NIC Hostmaster by electronic mail at the above address, or by phone at (415) 859-5539 or (800) 235-3155.

NOTE: The key people must have electronic mailboxes and NIC "handles," unique NIC database identifiers. If you have access to "WHOIS", please check to see if you are registered and if so, make sure the information is current. Include only your handle and any changes (if any) that need to be made in your entry. If you do not have access to "WHOIS", please provide all the information indicated and a NIC handle will be assigned.

(1) The name of the top-level domain to join.

For example: COM

(2) The NIC handle of the administrative head of the organization. Alternately, the person's name, title, mailing address, phone number, organization, and network mailbox. This is the contact point for administrative and policy questions about the domain. In the case of a research project, this should be the principal investigator.

For example:

Administrator

Organization	The NetWorthy Corporation
Name	Penelope Q. Sassafrass
Title	President
Mail Address	The NetWorthy Corporation 4676 Andrews Way, Suite 100 Santa Clara, CA 94302-1212
Phone Number	(415) 123-4567

NIC Handle PQS

(3) The NIC handle of the technical contact for the domain. Alternately, the person's name, title, mailing address, phone number, organization, and network mailbox. This is the contact point for problems concerning the domain or zone, as well as for updating information about the domain or zone.

For example:

Technical and Zone Contact

Organization	The NetWorthy Corporation
Name	Ansel A. Aardvark
Title	Executive Director
Mail Address	The NetWorthy Corporation 4676 Andrews Way, Suite 100 Santa Clara, CA. 94302-1212
Phone Number	(415) 123-6789
Net Mailbox	Aardvark@ECHO.TNC.COM
NIC Handle	AAA2

(4) The name of the domain (up to 12 characters). This is the name that will be used in tables and lists associating the domain with the domain server addresses. [While, from a technical standpoint, domain names can be quite long (programmers beware), shorter names are easier for people to cope with.]

For example: TNC

(5) A description of the servers that provide the domain service for translating names to addresses for hosts in this domain, and the date they will be operational.

A good way to answer this question is to say "Our server is supplied by person or company X and does whatever their standard issue server does."

For example: Our server is a copy of the one operated by the NIC; it will be installed and made operational on 1 November 1987.

(6) Domains must provide at least two independent servers for the domain. Establishing the servers in physically separate locations

and on different PSNs is strongly recommended. A description of the server machine and its backup, including

(a) Hardware and software (using keywords from the Assigned Numbers RFC).

(b) Host domain name and network addresses (which host on which network for each connected network).

(c) Any domain-style nicknames (please limit your domain-style nickname request to one)

For example:

- Hardware and software

  VAX-11/750 and UNIX, or
  IBM-PC and MS-DOS, or
  DEC-1090 and TOPS-20

- Host domain names and network addresses

  BAR.FOO.COM 10.9.0.193 on ARPANET

- Domain-style nickname

  BR.FOO.COM (same as BAR.FOO.COM 10.9.0.13 on ARPANET)

(7) Planned mapping of names of any other network hosts, other than the server machines, into the new domain's naming space.

For example:

BAR-FOO2.ARPA (10.8.0.193) -> FOO2.BAR.COM
BAR-FOO3.ARPA (10.7.0.193) -> FOO3.BAR.COM
BAR-FOO4.ARPA (10.6.0.193) -> FOO4.BAR.COM

(8) An estimate of the number of hosts that will be in the domain.

(a) Initially
(b) Within one year
(c) Two years
(d) Five years.

For example:

(a) Initially = 50
(b) One year = 100
(c) Two years = 200
(d) Five years = 500

(9) The date you expect the fully qualified domain name to become the official host name in HOSTS.TXT.

Please note: Registration of this domain does not imply an automatic name change to previously registered ARPANET or MILNET hosts that will be included in this domain. If changing to a fully qualified domain name (e.g., FOO.BAR.COM) causes a change in the official host name of an ARPANET or MILNET host, DCA approval must be obtained. This should be done after your domain name is approved by Hostmaster. Allow 10 working days for your requested changes to be processed. ARPANET (network 10) sites should contact ARPANETMGR@DDN1.ARPA. MILNET (network 26) sites should contact MILNETMGR@DDN1.ARPA.

(10) Please describe your organization briefly.

For example: The NetWorthy Corporation is a consulting organization of people working with UNIX and the C language in an electronic networking environment. It sponsors two technical conferences annually and distributes a bimonthly newsletter.

Appendix C

This appendix lists the papers, articles, and RFCs associated with the BIND service that you may find useful. You can obtain the RFCs online by using the ftp command as shown in Appendix B. See ftp(1c) in the ULTRIX Reference Pages for further information.

[Dunlap 86a] Dunlap, K. J., Bloom, J. M., "Experiences Implementing BIND, A Distributed Name Server for the DARPA Internet", Proceedings USENIX Summer Conference, Atlanta, Georga. June 1986, pages 172-181

[Dunlap 86b] Dunlap, K. J., "Name Server Operations Guide for BIND", Unix System Manager's Manual, SMM-11. 4.3 Berkeley Software Distribution, Virtual VAX-11 Version. University of California. April 1986

[Dyer 87] Dyer, S., and F. Hsu, "Hesiod", Project Athena Technical Plan - Name Service, April 1987, version 1.9.

[IEN-116] Postel J., "Internet Name Server", IEN-116, USC/Information Sciences Institute, August 1979.

[Mockapetris 88] Mockapetris, P. V., Dunlap, K. J., "Development of the Domain Name System", Proceedings ACM SIGCOMM 1988 Symposium, Stanford University, Stanford, California, August 1988.

[Quarterman 86] Quarterman, J., and J. Hoskins, "Notable Computer Networks",Communications of the ACM, October 1986, volume 29, number 10.

[RFC-882] P. Mockapetris, "Domain names - Concepts and Facilities," RFC-882, USC/Information Sciences Institute, November 1983.

[RFC-883] P. Mockapetris, "Domain names - Implementation and Specification," RFC-883, USC/Information Sciences Institute, November 1983.

[RFC-920] J. Postel and J. Reynolds, "Domain Requirements", RFC-920, USC/Information Sciences Institute October 1984.

[RFC-973] P. Mockapetris, "Domain System Changes and Observations", RFC-973, USC/Information Sciences Institute, January 1986.

[RFC-974] C. Partridge, "Mail routing and the domain system", RFC-974, CSNET CIC BBN Labs, January 1986.

[RFC-1031] W. Lazear, "MILNET Name Domain Transition", RFC-1031, November 1987.

[RFC-1032] M. K. Stahl, "Establishing a Domain - Guidelines for Administrators", RFC-1032, November 1987.

[RFC-1033] M. K. Lottor, "Domain Administrators Operations Guide", RFC-1033, November 1987.

[RFC-1034] Mockapetris, P. V., "Domain Names - Concepts and Facilities" RFC 1034, USC/Information Sciences Institute, November 1987.

[RFC-1035] Mockapetris, P. V., "Domain names - Implementation and Specification," RFC 1035, USC/Information Sciences Institute, November 1987.

Note

In the references listed, *RFC* refers to papers in the ARPA Request for Comments series and *IEN* refers to ARPA Internet Experiment Notes. Both the RFCs and IENs may be obtained from the Network Information Center, SRI International, Menlo Park, CA 94025, or from the authors of the papers.

Appendix D

This appendix provides sample interactive sessions with the nslookup command. These samples are intended to help you get started using the nslookup command. Here are the tasks shown in this appendix:

- Getting nslookup help
- Seeing which nslookup options are set
- Listing hosts in a domain
- Finding mail exchangers
- Finding the start of authority (SOA)
- Finding servers for a domain
- Obtaining a debug trace

D.1 Getting nslookup Help

To see a list of the nslookup commands, type a question mark (?) at the nslookup prompt:

```
# nslookup
Default Server:  wepel.cities.dec.com
Address:  0.0.0.0

> ?
Commands:          (identifiers are shown in uppercase, [] means optional)
NAME               - print info on host/domain NAME using default server
NAME1 NAME2        - as above, but use NAME2 as server
help or ?          - print help information
set OPTION         - set an option
    all            - print options, current server and host
    ALL            - print options, current server and host, state info
    [no]debug      - print debugging information
    [no]d2         - print exhaustive debugging information
    [no]defname    - append domain name to each query
    [no]recurse    - ask for recursive answer to query
    [no]vc         - always use a virtual circuit
    domain=NAME    - set default domain name to NAME
```

(continued on next page)

```
    root=NAME       - set root server to NAME
    retry=X         - set number of retries to X
    timeout=X       - set time-out interval to X
    querytype=X     - set query type to A,CNAME,HINFO,MB,MG,MINFO,MR,MX
    type=X          - set query type to A,CNAME,HINFO,MB,MG,MINFO,MR,MX
server NAME         - set default server to NAME, using default server
lserver NAME        - set default server to NAME, using initial server
finger [NAME]       - finger the optional NAME
root                - set current default server to the root
ls [-adhms] DOMAIN [> FILE] - list DOMAIN, optional output to FILE
                    -a = list CNAME entries
                    -d = list all entries
                    -h = list HINFO entries
                    -m = list MX entries
                    -s = list WKS entries
view FILE           - sort an 'ls' output file and view it with more
```

D.2 Seeing Which nslookup Options Are Set

To see which nslookup options are set, use the set all command:

```
# nslookup
Default Server:  wepel.cities.dec.com
Address:  0.0.0.0

> set all
Default Server:  wepel.cities.dec.com
Address:  0.0.0.0

Set options:
  debug         defname       search   recurse       novc
  querytype=A   class=IN      timeout=4     retry=4
  domain=cities.dec.com
  search list: cities.dec.com dec.com
  root=sri-nic.arpa
```

D.3 Listing Hosts in a Domain

The following example shows how to use the nslookup command to create a file listing the hosts in the domain cities.dec.com, and to then view that file:

```
# nslookup
Default Server:  wepel.cities.dec.com
Address:  0.0.0.0
```

(continued on next page)

```
> ls cities.dec.com > filename
[wepel.cities.dec.com]
##########
Received 531 records.
> view filename
amherst                         128.67.45.1
ayers                           128.67.42.2
berlin                          128.67.45.3
boston                          128.67.45.4
cannes                          128.67.45.5
chandler                        128.67.45.6
chicago                         128.67.45.7
denver                          128.67.46.8
galway                          128.67.45.9
hollis                          128.67.49.10
ipswich                         128.67.45.11
laconia                         128.67.48.12
london                          128.67.45.13
madrid                          128.67.45.14
mason                           128.67.45.15
milford                         128.67.46.16
nashua                          128.67.45.17
newyork                         128.67.45.18
--More-- <RETURN>
paris                           128.67.42.19
phoenix                         128.67.46.20
tempe                           128.67.45.21
temple                          128.67.45.22
wilton                          128.67.45.23
<CTRL/c>
> <CTRL/d>
#
```

D.4 Finding Mail Exchangers

The following example shows how to use the nslookup command to find the mail exchanger for any system in the domain wepel.cities.dec.com. Note the use of a bogus host name. In the following example, the bogus host name is nohost:

```
# nslookup
Default Server:  wepel.cities.dec.com
Address:  0.0.0.0

> set type=mx
> nohost
Server:  wepel.cities.dec.com
Address:  0.0.0.0
```

(continued on next page)

```
nohost.cities.dec.com pref = 51, mail exchanger = noun.cities.dec.com
nohost.cities.dec.com pref = 50, mail exchanger = wepel.cities.dec.com
noun.cities.dec.com   inet address = 128.45.45.79
wepel.cities.dec.com  inet address = 128.45.45.93
> wepel
Server:  wepel.cities.dec.com
Address:  0.0.0.0

cities.dec.com    origin = wepel.cities.dec.com
     mail addr = doe.wepel.cities.dec.com
     serial=10, refresh=1800, retry=3600, expire=1209600, min=86400
```

D.5 Finding the Start of Authority

The following sample session shows how to use the nslookup command to find the start of authority for the hosts named wepel and decwrl.dec.com:

```
# nslookup
Default Server:  wepel.cities.dec.com
Address:  0.0.0.0

> set type=SOA
> wepel
Server:  wepel.cities.dec.com
Address:  0.0.0.0

cities.dec.com    origin = wepel.cities.dec.com
     mail addr = doe.wepel.cities.dec.com
     serial=10, refresh=1800, retry=3600, expire=1209600, min=86400
> decwrl.dec.com.
Server:  wepel.cities.dec.com
Address:  0.0.0.0

dec.com  origin = decwrl.dec.com
     mail addr = postmaster.decwrl.dec.com
     serial=197, refresh=43200, retry=3600, expire=1209600, min=86400
> <CTRL/d>
```

D.6 Finding Servers for a Domain

The following example shows how to use the nslookup command to find the servers for the domain mit.edu.:

```
# nslookup
Default Server:  wepel.cities.dec.com
Address:  0.0.0.0
```

(continued on next page)

```
> server sri-nic.arpa.
Default Server:  sri-nic.arpa
Address:  26.0.0.73

> set domain=mit.edu.
> ls
Server:  sri-nic.arpa
Address:  26.0.0.73

Name:     ls.mit.edu.
Served by:
- MIT-STRAWB.ARPA
        18.71.0.151
        MIT.EDU
- W20NS.MIT.EDU
        18.70.0.160
        MIT.EDU
- BITSY.MIT.EDU
        18.72.0.3
        MIT.EDU
- LITHIUM.LCS.MIT.EDU
        18.26.0.121
        MIT.EDU
> <CTRL/d>
#
```

D.7 Obtaining a Debug Trace

The following example shows how to use the nslookup command to help debug the BIND service:

```
# nslookup
Default Server:  wepel.cities.dec.com
Address:  0.0.0.0

> set debug
> set d2
> foobar
Server:  wepel.cities.dec.com
Address:  0.0.0.0

res_mkquery(0, foobar.cities.dec.com, 1, 1)
------------
SendRequest()
    HEADER:
    opcode = QUERY, id = 1, rcode = NOERROR
    header flags:  query, want recursion
    questions = 1,  answers = 0,  n.s. = 0,  additional = 0
```

(continued on next page)

```
    QUESTIONS:
     foobar.cities.dec.com, type = A, class = IN

------------
------------
Got answer:
    HEADER:
     opcode = QUERY, id = 1, rcode = NOERROR
     header flags:  resp, auth. answer, want recursion, recursion avail.
     questions = 1,  answers = 0,  n.s. = 0,  additional = 1

    QUESTIONS:
     foobar.cities.dec.com, type = A, class = IN
    ADDITIONAL RECORDS:
    ->  cities.dec.com
     type = SOA, class = IN, ttl = 86400, dlen = 37
     origin = wepel.cities.dec.com
     mail addr = doe.wepel.cities.dec.com
     serial=10, refresh=1800, retry=3600, expire=1209600, min=86400

------------
Name:    foobar.cities.dec.com

> noun
Server:  wepel.cities.dec.com
Address:  0.0.0.0

res_mkquery(0, noun.cities.dec.com, 1, 1)
------------
SendRequest()
    HEADER:
     opcode = QUERY, id = 2, rcode = NOERROR
     header flags:  query, want recursion
     questions = 1,  answers = 0,  n.s. = 0,  additional = 0

    QUESTIONS:
     noun.cities.dec.com, type = A, class = IN

------------
------------
Got answer:
    HEADER:
     opcode = QUERY, id = 2, rcode = NOERROR
     header flags:  resp, auth. answer, want recursion, recursion avail.
     questions = 1,  answers = 1,  n.s. = 0,  additional = 0

    QUESTIONS:
     noun.cities.dec.com, type = A, class = IN
    ANSWERS:
```

(continued on next page)

```
     ->  noun.cities.dec.com
     type = A, class = IN, ttl = 86400, dlen = 4
     inet address = 128.45.45.79

------------
Name:    noun.cities.dec.com
Address:  128.45.45.79

> set type=SOA
> noun
Server:  wepel.cities.dec.com
Address:  0.0.0.0

res_mkquery(0, noun.cities.dec.com, 1, 6)
------------
SendRequest()
    HEADER:
     opcode = QUERY, id = 3, rcode = NOERROR
     header flags:  query, want recursion
     questions = 1,  answers = 0,  n.s. = 0,  additional = 0

    QUESTIONS:
     noun.cities.dec.com, type = SOA, class = IN

------------
------------
Got answer:
    HEADER:
     opcode = QUERY, id = 3, rcode = NOERROR
     header flags:  resp, auth. answer, want recursion, recursion avail.
     questions = 1,  answers = 0,  n.s. = 0,  additional = 1

    QUESTIONS:
     noun.cities.dec.com, type = SOA, class = IN
    ADDITIONAL RECORDS:
    ->  cities.dec.com
     type = SOA, class = IN, ttl = 86400, dlen = 37
     origin = wepel.cities.dec.com
     mail addr = doe.wepel.cities.dec.com
     serial=10, refresh=1800, retry=3600, expire=1209600, min=86400

------------
cities.dec.com
     type = SOA, class = IN, ttl = 86400, dlen = 37
     origin = wepel.cities.dec.com
     mail addr = doe.wepel.cities.dec.com
     serial=10, refresh=1800, retry=3600, expire=1209600, min=86400
```

(continued on next page)

```
> decwrl.dec.com.
Server:  wepel.cities.dec.com
Address:  0.0.0.0

res_mkquery(0, decwrl.dec.com, 1, 6)
------------
SendRequest()
    HEADER:
    opcode = QUERY, id = 4, rcode = NOERROR
    header flags:  query, want recursion
    questions = 1,  answers = 0,  n.s. = 0,  additional = 0

    QUESTIONS:
    decwrl.dec.com, type = SOA, class = IN

------------
------------
Got answer:
    HEADER:
    opcode = QUERY, id = 4, rcode = NOERROR
    header flags:  resp, auth. answer, want recursion, recursion avail.
    questions = 1,  answers = 0,  n.s. = 0,  additional = 1

    QUESTIONS:
    decwrl.dec.com, type = SOA, class = IN
    ADDITIONAL RECORDS:
    ->  dec.com
    type = SOA, class = IN, ttl = 83633, dlen = 35
    origin = decwrl.dec.com
    mail addr = postmaster.decwrl.dec.com
    serial=197, refresh=43200, retry=3600, expire=1209600, min=86400

------------
dec.com
    type = SOA, class = IN, ttl = 83633, dlen = 35
    origin = decwrl.dec.com
    mail addr = postmaster.decwrl.dec.com
    serial=197, refresh=43200, retry=3600, expire=1209600, min=86400
> <CTRL/d>
#
```

Index

C

D

F

G

W

Z

HOW TO ORDER ADDITIONAL DOCUMENTATION

DIRECT TELEPHONE ORDERS

In Continental USA
and New Hampshire,
Alaska or Hawaii
call **800-DIGITAL**

In Canada
call **800-267-6215**

DIRECT MAIL ORDERS (U.S. and Puerto Rico*)

DIGITAL EQUIPMENT CORPORATION
P.O. Box CS2008
Nashua, New Hampshire 03061

DIRECT MAIL ORDERS (Canada)

DIGITAL EQUIPMENT OF CANADA LTD.
100 Herzberg Road
Kanata, Ontario K2K 2A6
Attn: Direct Order Desk

INTERNATIONAL

DIGITAL EQUIPMENT CORPORATION
PSG Business Manager
c/o Digital's local subsidiary
or approved distributor

Internal orders should be placed through the Software Distribution Center (SDC), Digital Equipment Corporation, Westminster, Massachusetts 01473

*Any prepaid order from Puerto Rico must be placed
with the Local Digital Subsidiary:
809–754–7575

Reader's Comments

Note: This form is for document comments only. DIGITAL will use comments submitted on this form at the company's discretion. If you require a written reply and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Did you find this manual understandable, usable, and well-organized? Please make suggestions for improvement. _____

_____

_____

_____

Did you find errors in this manual? If so, specify the error and the page number.

_____

_____

_____

Please indicate the type of user/reader that you most nearly represent.

- ☐ Assembly language programmer
- ☐ Higher-level language programmer
- ☐ Occasional programmer (experienced)
- ☐ User with little programming experience
- ☐ Student programmer
- ☐ Other (please specify) _____

Name _____ Date _____

Organization _____

Street _____

Do Not Tear - Fold Here and Tape

digital

No Postage
Necessary
if Mailed in the
United States

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO.33 MAYNARD MASS.

POSTAGE WILL BE PAID BY ADDRESSEE

Digital Equipment Corporation
Documentation Manager
ULTRIX Documentation Group
ZKO3-3/X18
Spit Brook Road
Nashua, N.H.
03063

Do Not Tear - Fold Here and Tape

Cut Along Dotted Line

Reader's Comments

Note: This form is for document comments only. DIGITAL will use comments submitted on this form at the company's discretion. If you require a written reply and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Did you find this manual understandable, usable, and well-organized? Please make suggestions for improvement. _____

_____

_____

_____

Did you find errors in this manual? If so, specify the error and the page number.

_____

_____

_____

Please indicate the type of user/reader that you most nearly represent.

- ☐ Assembly language programmer
- ☐ Higher-level language programmer
- ☐ Occasional programmer (experienced)
- ☐ User with little programming experience
- ☐ Student programmer
- ☐ Other (please specify) _____

Name _____ Date _____

Organization _____

Street _____

Do Not Tear - Fold Here and Tape

digital

No Postage
Necessary
if Mailed in the
United States

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO.33 MAYNARD MASS.

POSTAGE WILL BE PAID BY ADDRESSEE

Digital Equipment Corporation
Documentation Manager
ULTRIX Documentation Group
ZKO3-3/X18
Spit Brook Road
Nashua, N.H.
03063

Do Not Tear - Fold Here and Tape

Cut Along Dotted Line