VAX Distributed Name Service

Management Guide

Order No. AA–KN85B–TE

August 1988

This manual describes the VAX Distributed Name Service and tells you how to plan and manage it.

Supersession/Update Information:	Revised for new covers.
Operating System and Version:	VAX/VMS V4.6–5.0
Software Version:	VAX Distributed Name Service V1.1

digital

AA-KN85B-TE
First Edition, August 1988

The information in this document is subject to change without notice and should not be construed as a commitment by Digital Equipment Corporation. Digital Equipment Corporation assumes no responsibility for any errors that may appear in this document.

The software described in this document is furnished under a license and may only be used or copied in accordance with the terms of such license.

No responsibility is assumed for the use or reliability of software on equipment that is not supplied by Digital Equipment Corporation or its affiliated companies.

The postage-prepaid Reader's Comments form on the last page of this document requests the user's critical evaluation to assist us in preparing future documentation.

This manual was produced by Networks and Communications Publications.

Contents

FIGURES

TABLES

Preface

The VAX Distributed Name Service (DNS) provides you with a networkwide means of assigning unique, location-independent names to network resources, also known as objects. Objects can be files, disks, nodes, queues, mailboxes, and so forth. A DNS object name is known and updated networkwide, so that all users and applications in your network can access an object using the same, unique name. Because DNS names are location independent, users and applications do not need to know on which node an object resides. DNS translates an object's name to a set of attributes; its network address is one of these attributes.

Applications that are designed to work with DNS use DNS names to name their objects. For example, DNS is currently a prerequisite for the Remote System Manager (RSM) V2.0 and the VAX Distributed File Service (DFS), both of which use DNS names for their objects.

Over time, additional Digital applications will use DNS to name their objects. In DECnet Phase V networks, node names will be DNS names that are stored on DNS server nodes.

This manual describes DNS concepts and tells you how to plan and manage your DNS namespace.

Intended Audience

This manual is intended for anyone who will install and manage DNS in their network. If your network is managed or configured by a network administrator, that person should read this manual to understand the networkwide planning requirements for DNS.

Chapters 1 and 2 are of interest to anyone who wants a general understanding of DNS.

Structure of This Document

This manual contains the following chapters and appendixes:

- Chapter 1 introduces key DNS concepts.
- Chapter 2 provides guidelines for planning the installation of DNS in your network. This includes whether to create a single-directory or a hierarchical namespace. After you read Chapter 2, you should be ready to install DNS.
- Chapter 3 tells you how to use the DNS$CONTROL management program to manage DNS.
- Chapter 4 summarizes the DNS$CONTROL commands, including their functions and formats.
- Appendix A alphabetically lists messages that you might receive while running DNS and tells you how to deal with them.
- Appendix B provides information on attributes associated with entries in the DNS namespace.
- Appendix C provides information on messages that may be logged during DNS operation and tells you how to deal with them.
- Appendix D summarizes valid characters that can be included in DNS names.

The glossary at the end of the manual defines DNS terms.

Associated Documents

Have the following additional documents available for reference:

- *VAX Distributed Name Service Installation Guide*
- *VAX/VMS Network Control Program Reference Manual*

Conventions Used in This Document

`Special type` — This special type in examples indicates system output or user input.

Red type — Red type in examples indicates user input.

Boldface type — Boldface type flags new terms and concepts.

RET — Press the RETURN key.

UPPERCASE — Uppercase letters in command lines indicate words that must be entered. You can abbreviate command keywords to the first three characters or the minimum unique abbreviation.

lowercase italics — Lowercase italics in command syntax or examples indicate variables for which either the user or the system supplies a value.

[ ] — Square brackets in command lines indicate that the enclosed values are optional. (Do not type the brackets.)

{ } — Braces in command lines indicate that you must specify one (and only one) of the enclosed values. (Do not type the braces.)

CTRL/*x* — Hold down the CONTROL key and then press the key specified by *x*.

Chapter 1
Understanding DNS

This chapter is structured so that you must read it from beginning to end in order to gain a clear and complete understanding of the VAX Distributed Name Service.

1.1 What is DNS?

The VAX Distributed Name Service (DNS) provides you with a networkwide means of assigning unique, location-independent names to network resources, also known as **objects**. Objects can be files, disks, nodes, queues, mailboxes, and so forth. A DNS **object name** is known and updated networkwide, so that all users and applications in your network can access an object using the same, unique name. Because DNS names are location-independent, users and applications do not need to know on which node an object resides. DNS translates an object's name to a set of **attributes**; its network address is one of these attributes.

Applications that are designed to work with DNS use DNS names to name their objects. For example, DNS is a prerequisite for the following applications:

- Remote System Manager (RSM) V2.0
- VAX Distributed File Service (DFS)

DFS uses DNS names to name its file access points, and RSM uses DNS names to name its client nodes. Over time, additional Digital applications will use DNS to name their objects. In DECnet Phase V networks, node names will be DNS names that are stored on DNS server nodes.

1.1.1 DNS Servers and Clients

DNS is based on a client-server design, in which two cooperating components of software, the server and the client, work together to make the service function.

You install DNS software on a **DNS server** node. A DNS server is any node in your network on which DNS **object names** are stored. You can have as many DNS servers in your network as you want. (Chapter 2 provides guidelines for choosing the number of DNS servers in your network and for best locating those servers in your network.)

All applications that use DNS, such as RSM and DFS, contain DNS **client software**. Client software enables users of those applications to name their objects using DNS names. These names are stored on DNS server nodes. Users can then use DNS names to access their objects. When such an object is accessed, the DNS **server software** translates the object's name into a set of **attributes**; its network address is one of these attributes.

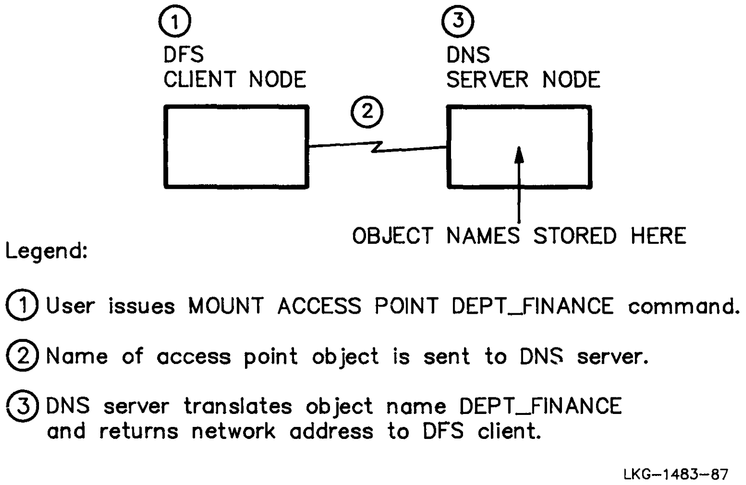
DNS clients and servers use DECnet to communicate.

1.1.2 The Function of DNS

Figure 1-1 shows the function of DNS, in which DFS is the application that is using DNS. In Figure 1-1, a user on a DFS client node issues a MOUNT ACCESS POINT DEPT_FINANCE command, where DEPT_FINANCE is the name of the access point. The access point's name, which is stored on the DNS server, is sent to the DNS server and translated there. The resulting name information, which includes the network address of the access point, is returned to the DFS client node. The DFS client software now knows the address in the network where the access point DEPT_FINANCE is located. The DFS client software establishes a connection to that network address and sends the MOUNT ACCESS POINT request.

The example in Figure 1-1 shows the primary purpose of DNS: to translate location-independent, networkwide object names into a set of attributes for an object. One of the most important attributes is an object's network address. (Appendix B summarizes DNS object attributes.)

Figure 1-1: An Example of DNS Function


①
DFS
CLIENT NODE
②
③
DNS
SERVER NODE
OBJECT NAMES STORED HERE
Legend:
① User issues MOUNT ACCESS POINT DEPT_FINANCE command.
② Name of access point object is sent to DNS server.
③ DNS server translates object name DEPT_FINANCE and returns network address to DFS client.
LKG–1483–87

1.1.3 Benefits of DNS

Because DNS names are location independent, DNS makes your physical network transparent to both applications and users. Users and applications do not need to know on which node in your network an object resides. Also, with DNS you create a single, networkwide **namespace** in which all object names in your network are stored. This namespace is easily managed, whether your network consists of a half dozen nodes or thousands of nodes. A networkwide namespace also enables you to standardize object naming practices throughout your network.

Whenever you create a DNS object name, that name is guaranteed to be unique throughout your network. Whenever you update an object's attributes, which are stored by DNS, the new information for the object is automatically made known to all the appropriate DNS servers in your network.

1.2 How DNS Naming Practices Differ from Traditional Practices

This section discusses the differences between traditional ways of naming objects and distributed naming with DNS. Traditionally, object names have been arbitrarily assigned on each node in a network. With DNS, object names are assigned on DNS server nodes and are consistently known and updated on those servers.

1.2.1 Characteristics of Traditional Naming Practices

Traditionally, objects have been arbitrarily named on each node in a network. In this environment, objects can be accessed over the network only by including location-specific information in their names. For example, the name of a disk volume on a VMS node would be accessed by specifying its node name and the device name (or logical name) on that system. With these naming practices, users would refer to volume PROJ$DISK on node BARN as:

```
BARN::PROJ$DISK:
```

The following characterizes traditional naming practices:

- In order to access an object over the network, users must know the exact location (node name) of the object in the network. They must also find out the object's modified name if it is moved to another node in the network.
- The assignment of object names is arbitrary and unsystematic on a networkwide basis.
- Users throughout the network might know the same object by different names.
- Efforts to manage object names are unorganized and difficult.
- Access rights to objects are arbitrarily assigned on each node in a network.

1.2.2 Characteristics of Distributed Naming with DNS

The following characterizes distributed naming with DNS:

- Object names are uniquely and consistently known networkwide. All users throughout the network can access an object using the same, unique name.
- Objects can be relocated in the network without requiring applications or users to change the names they use to refer to them.
- The process by which the network addresses of new or relocated objects are propagated throughout the network is automated.
- DNS creates a single, networkwide namespace in which all object names in the network are stored. This makes the network and its component systems easier to manage.
- Access rights to object names are defined networkwide.

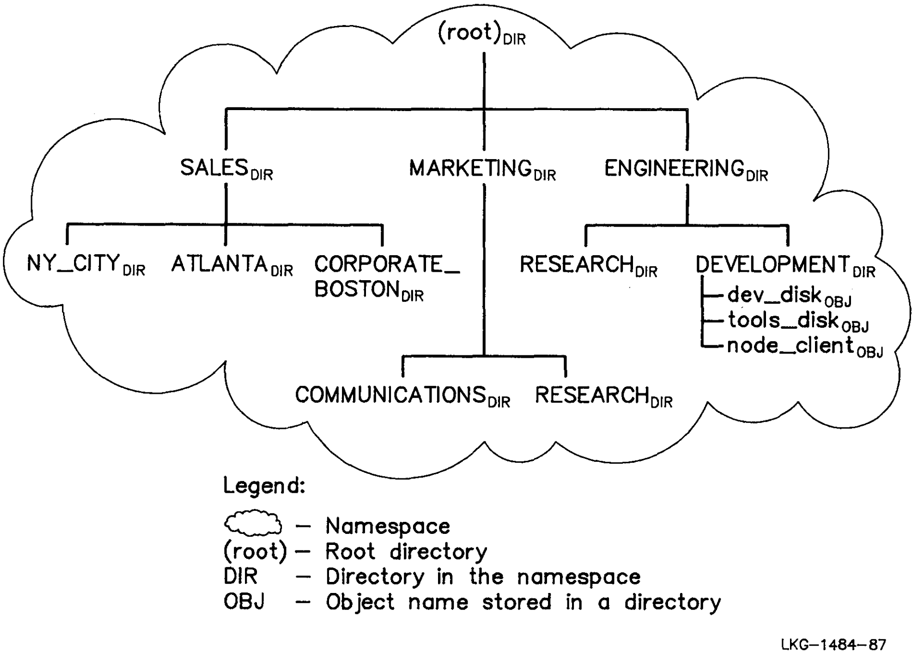
1.3 The DNS Namespace

The DNS **namespace** is not a physical entity. It is the logical structure in which the names of all **objects** in your network are stored. In the namespace, **object names** are stored in DNS **directories** that are hierarchically structured, much like VMS directories and subdirectories.

The main directory in the namespace is called the **root directory**. You can store all object names in the root directory, or you can build a hierarchy of DNS directories under the root directory. These namespace structures are described in more detail in Section 1.4.

Figure 1-2 shows a namespace and the hierarchy of directories that comprise it.

Figure 1-2: A Hierarchical Namespace


(root)$_{\text{DIR}}$
SALES$_{\text{DIR}}$
MARKETING$_{\text{DIR}}$
ENGINEERING$_{\text{DIR}}$
NY_CITY$_{\text{DIR}}$
ATLANTA$_{\text{DIR}}$
CORPORATE_ BOSTON$_{\text{DIR}}$
RESEARCH$_{\text{DIR}}$
DEVELOPMENT$_{\text{DIR}}$
dev_disk$_{\text{OBJ}}$
tools_disk$_{\text{OBJ}}$
node_client$_{\text{OBJ}}$
COMMUNICATIONS$_{\text{DIR}}$
RESEARCH$_{\text{DIR}}$
Legend:
☁ – Namespace
(root) – Root directory
DIR – Directory in the namespace
OBJ – Object name stored in a directory
LKG–1484–87

The namespace is automatically created when you install DNS software on the first DNS server node in your network.

There should be one namespace per network. It is possible to have more than one namespace in a network, but DNS does not allow namespaces to communicate with each other.

1.4 Structuring the Namespace

You can create your namespace using one of two structures:

- The **single-directory namespace** structure, in which you store all object names in the **root directory**.
- The **hierarchical namespace** structure, in which you store object names in a hierarchy of directories that you create under the root directory.

These structures are described in the following sections.

Designing and structuring your namespace requires careful planning on a networkwide basis. If your network is managed or configured by a network administrator, that person should participate in structuring your namespace. Chapter 2 provides guidelines for structuring the namespace.

1.4.1 The Single-Directory Namespace

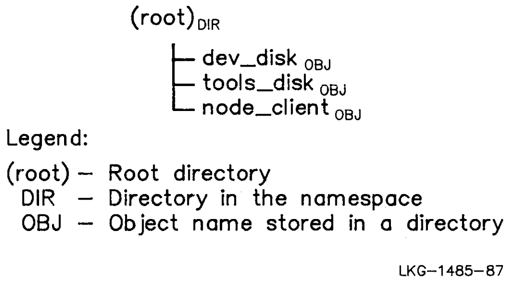
A single-directory namespace consists of one directory, the root directory. In a single-directory namespace, you store all object names in the root directory.

For ease of use, the DNS installation automatically creates the root directory. This provides you with a single-directory namespace. This allows you to store DFS and RSM object names in the root directory as soon as DNS installation is complete.

A single-directory namespace does not limit the number of DNS servers you can have in your network. You can still have as many servers as you want.

Figure 1-3 shows a single-directory namespace. In the figure, DFS has stored the object names dev_disk and tools_disk in the root directory. RSM has stored the object name node_client in the root directory.

Figure 1-3: A Single-Directory Namespace



1.4.2 The Hierarchical Namespace

In a hierarchical namespace, you create a hierarchy of directories under the root directory. You can store object names in any directory in the hierarchy. In a hierarchical namespace, the root directory is always the top directory in the hierarchy.

Figure 1-2 showed a hierarchical namespace. The hierarchical nature of the namespace allows virtually unlimited expansion of the number of directories in the namespace. The hierarchical structure stores object names in an orderly way as a network expands over time.

When you create the namespace shown in Figure 1-2, the directory ENGINEERING must exist in the namespace before you can create the directory ENGINEERING.DEVELOPMENT.

1.4.2.1 Choosing the Namespace Structure

For ease of use, the DNS installation automatically creates a single-directory namespace, (that is, the root directory). Therefore, you can store DFS and RSM object names in the namespace as soon as installation is complete. Choosing a single-directory namespace makes registering your object names with DNS quick and simple. If your network consists of fewer than 100 nodes, a single-directory namespace will probably satisfy your needs.

If you want to allow for future expansion of your network and its namespace, you might want to create a hierarchical namespace, as described in Chapter 2.

1.4.3 The Syntax of DNS Names

The syntax of DNS names reflects the structure of your namespace. In DNS, an object's name reflects the complete path through your namespace from the root directory to the object entry in a particular directory. Therefore, all DNS names begin with a **dot character** (.), which denotes the root directory.

In Figure 1-3, the object names stored in the namespace are expressed as follows:

- .DEV_DISK
- .TOOLS_DISK
- .NODE_CLIENT

NOTE

> If you do not include the initial dot character (.) when specifying a DNS object name, VMS attempts to translate the first name in the string using the VMS logical name facility. For example, in the DNS name ENGINEERING.DEVELOPMENT.DEV_DISK, VMS would attempt to translate the name ENGINEERING as a logical name. If the translation succeeds, the resulting name is passed to a DNS server.

In Figure 1-2, the object names stored in the directory named DEVELOPMENT are expressed as follows:

- .ENGINEERING.DEVELOPMENT.DEV_DISK
- .ENGINEERING.DEVELOPMENT.TOOLS_DISK
- .ENGINEERING.DEVELOPMENT.NODE_CLIENT

Note that in a DNS name, any string of characters to the left of the dot character is the name of a directory in the namespace.

1.4.3.1 The Namespace Name As Part of DNS Names

During the first DNS installation in your network, you are asked to provide a name for your namespace. The **namespace name** defaults to the DECnet node name of the server node on which you are installing DNS followed by _NS. For example, installing DNS on server node APPLE gives you a default namespace name of APPLE_NS.

Whenever DNS returns an object name, the object name is always preceded by its namespace name followed by a colon (:).

For example, if the object name .ENGINEERING.DEVELOPMENT.DEV_DISK is stored in a namespace named APPLE_NS, then DNS displays that object name as follows:

APPLE_NS:.ENGINEERING.DEVELOPMENT.DEV_DISK

1.4.3.2 Valid Characters for DNS Names

Appendix D summarizes the characters that a DNS name can contain.

1.5 Distributing the Namespace

This section introduces the concepts involved in distributing your namespace on multiple DNS servers in your network. Distributing the namespace increases the availability of DNS object names and improves DNS performance. Chapter 2 provides guidelines for distributing your namespace.

1.5.1 Distributing the Namespace to Increase Name Availability

To increase the availability of DNS object names, you can store all or part of your namespace on multiple servers in your network. Doing so increases the likelihood that the object name can be translated by DNS, even if one or more DNS servers are temporarily unavailable. You should choose carefully which nodes in your network will be DNS servers. DNS servers should be stable systems that are not scheduled for frequent downtime.

1.5.2 Distributing the Namespace to Improve DNS Performance

To improve DNS performance, you can make all or part of your namespace available on several DNS servers. The portions of your namespace that are accessed frequently by users and applications in a particular section of your network can be stored on DNS servers that are located in that section of your network. This improves DNS response time when a user or an application specifies a DNS object name that must be sent over the network to a DNS server.

1.5.3 Concepts of Namespace Distribution

Guidelines for partitioning and replicating all or part of your namespace are described in Chapter 2.

Before you can successfully distribute your namespace, you must understand the key concepts involved in namespace distribution. In DNS, the **directory** is the unit by which you **partition** and **replicate** the namespace. A collection of directories stored on a DNS server node is called a **clearinghouse**. There is at least one clearinghouse on each DNS server node in your network. You distribute your namespace by first making **read-only copies** of **master directories** and then placing them in clearinghouses on DNS servers throughout your network.

The following sections describe clearinghouses, master directories, and read-only directories in more detail.

1.5.3.1 Clearinghouses

A **clearinghouse** contains a set of DNS directories in the namespace. A clearinghouse is automatically created when you install a DNS server. On a DNS server, DNS creates a VMS file in the [DNS$SERVER] directory that contains information on the directories in that server's clearinghouse.

A clearinghouse is a named object in the namespace. A **clearinghouse object name** is created during installation. DNS clients must know about clearinghouse objects in order to locate a directory in the namespace. Clearinghouse objects are stored in directories that have been allowed to contain them, as described in Chapter 3.

During DNS installation, you are asked to supply a name for the clearinghouse being created. The clearinghouse name defaults to the DECnet node name of the server node followed by _CH. For example, if you install DNS on server node APPLE, the default name for the clearinghouse created on that server is APPLE_CH.

Because DNS installation creates a single-directory namespace, it stores a clearinghouse object name in the root directory by default. Therefore, in a single-directory namespace, the name of clearinghouse APPLE_CH is expressed as .APPLE_CH.

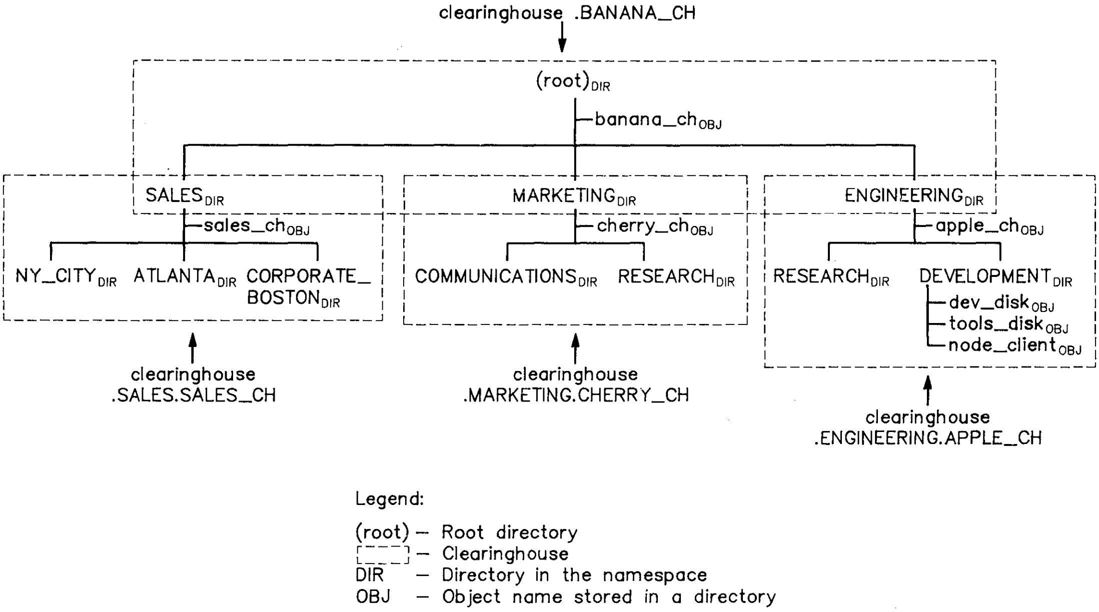
During installation in a hierarchical namespace, you must specify the directory in the namespace that will contain the clearinghouse object. For example, you could specify the clearinghouse object APPLE_CH as .ENGINEERING.APPLE_CH, which would store the clearinghouse object name in the directory ENGINEERING, as shown in Figure 1-4.

Figure 1-4 shows a hierarchical namespace that consists of four clearinghouses. The clearinghouses and the names of the DNS servers where they reside are as follows:

- Clearinghouse .BANANA_CH is on DNS server node BANANA.
- Clearinghouse .SALES.SALES_CH is on DNS server node SALES.
- Clearinghouse .MARKETING.CHERRY_CH is on DNS server node CHERRY.
- Clearinghouse .ENGINEERING.APPLE_CH is on DNS server node APPLE.

Table 1-1 lists the directories contained in each clearinghouse.

Figure 1-4: Clearinghouses and Their Component Directories



LKG-1486-87

Table 1-1: Clearinghouses and Their Component Directories (Figure 1-4)

Clearinghouse	Directories
.BANANA_CH	. (root) .sales .marketing .engineering
.SALES.SALES_CH	.sales .sales.ny_city .sales.atlanta .sales.corporate_boston
.MARKETING.CHERRY_CH	.marketing .marketing.communications .marketing.research
.ENGINEERING.APPLE_CH	.engineering .engineering.research .engineering.development

1.5.3.2 Distributing Read-Only Copies of Master Directories in Multiple Clearinghouses

A directory is the unit by which the namespace is paritioned and replicated. Directories are stored in clearinghouses on DNS server nodes. You distribute your namespace by first making read-only copies of master directories and then placing them in clearinghouses on DNS servers throughout your network.

In DNS, any directory that you create using the DNS$CONTROL CREATE DIRECTORY command, is a **master directory**. A master directory is a directory in which you can:

- Create object names
- Modify the attributes of object names

- Create a **child directory**

A **read-only directory** is a copy of a master directory that you make using the DNS$CONTROL COPY DIRECTORY command. You create read-only copies of master directories so you can distribute them throughout your namespace to increase DNS availability and to improve DNS performance.

You cannot update or modify the contents of a read-only directory. It can only be read by applications so that the object names stored there can be looked up and translated. In DNS, updates to a master directory are automatically propagated to all read-only copies of that master directory. Section 1.6 describes the DNS update function in detail.

Figure 1-5 shows a namespace having the same four clearinghouses as Figure 1-4. However, Figure 1-5 points out that the master directories .SALES, .MARKETING, and .ENGINEERING are stored in clearinghouse .BANANA_CH, and their read-only copies are stored in clearinghouses .SALES.SALES_CH, .MARKETING.CHERRY_CH, and .ENGINEERING.APPLE_CH, respectively. All of the other directories in Figure 1-5 are master directories.

Table 1-2 lists the directories contained in each clearinghouse and indicates whether each directory is a master directory or a read-only directory.

Figure 1-5: Distributing Read-only Copies of Master Directories in Multiple Clearinghouses

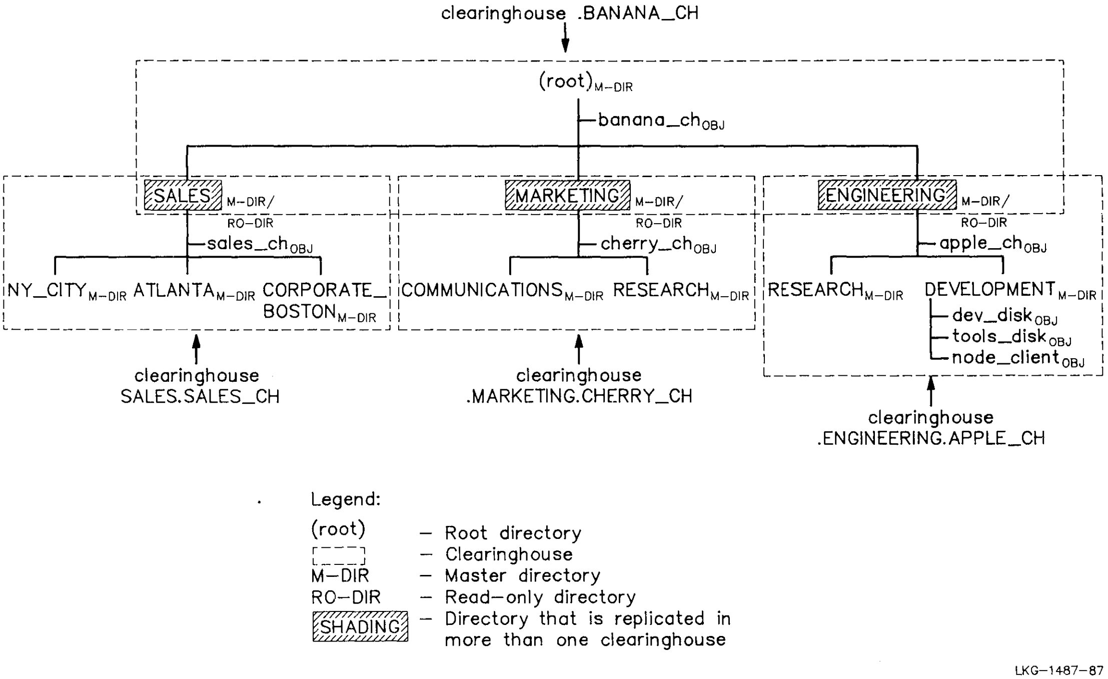


Table 1-2: Clearinghouses and Their Component Directories (Figure 1-5)

Clearinghouse	Directories
.BANANA_CH	(root) **(master)** .sales **(master)** .marketing **(master)** .engineering **(master)**
.SALES.SALES_CH	.sales **(read-only)** .sales.ny_city **(master)** .sales.atlanta **(master)** .sales.corporate_boston **(master)**
.MARKETING.CHERRY_CH	.marketing **(read-only)** .marketing.communications **(master)** .marketing.research **(master)**
.ENGINEERING.APPLE_CH	.engineering **(read-only)** .engineering.research **(master)** .engineering.development **(master)**

1.6 The DNS Update Function

A key function in DNS is the **update function**. This ensures that any changes (additions, deletions, or modifications) made to the contents of a master directory are automatically propagated to all its read-only copies.

DNS attempts to update all read-only directories at regular intervals. Each master directory has an attribute associated with it that stipulates its degree of **convergence**. Convergence refers to how frequently the DNS software executes the update function for a master directory. The DNS update function is also called a **skulk** operation. You will see the term skulk in various DNS displays.

The value of a master directory's convergence attribute controls the frequency with which DNS attempts to update its read-only copies. If the value of the convergence attribute is set to HIGH, DNS immediately attempts to update all read-only copies whenever a change is made to the master. If the update fails, DNS attempts the update every 12 hours until it succeeds. If the convergence value is set to LOW, DNS waits until the next scheduled update function to update all read-only copies of the master. By default, the value of the convergence attribute is HIGH.

DNS also allows you to force an update of a master directory by using the DNS$CONTROL UPDATE DIRECTORY command.

1.7 DNS Access Control

DNS enables you to control access to object names stored in the namespace. DNS allows you to grant five types of access:

1. Read
2. Write
3. Delete
4. Test
5. Control

These access rights are described in Chapter 3.

DNS installation grants all users read, write, delete, test, and control access to all object names in the namespace. After installation, you can alter access rights as you see fit. See Chapter 3 for more information.

1.8 Summary

The following summarizes the major points in this chapter.

- DNS enables applications, such as Remote System Manager (RSM) V2.0 and VAX Distributed File Service (DFS), to name their objects using location-independent names. These names are unique throughout the network and are consistently known on all DNS server nodes.

- Because DNS names are location independent, the resources they represent can be moved to any location in the network without users knowing about the change.
- DNS object names are stored on DNS server nodes.
- Applications that use DNS, such as RSM and DFS, contain DNS client software. This allows users of those applications to use DNS names to name their objects.
- DNS provides a single, networkwide namespace.
- Applications that use DNS store their object names in DNS directories that reside in the namespace.
- The main directory in the namespace is called the root directory.
- DNS installation creates the root directory and thus, a single-directory namespace. This enables you to store object names in the root directory as soon as installation is complete.
- You can also create a hierarchical namespace by creating a hierarchy of DNS directories under the root directory. This enables you to store object names in any directory in the hierarchy.
- The syntax of a DNS name reflects the path through your namespace from the root directory to an object entry in a particular directory.
- A clearinghouse contains a set of DNS directories in the namespace.
- If you install multiple DNS servers in the network, you can copy all or part of the namespace to any DNS server. Doing so improves the availability and performance of DNS.
- You distribute your namespace by placing read-only copies of master directories in clearinghouses on DNS servers throughout your network.
- Whenever you make a change to the contents of a master directory, DNS automatically propagates the new information to all read-only copies of the master directory.
- You can control the frequency with which DNS updates read-only directories by setting the value of the master directory's convergence attribute.
- DNS enables you to control access to object names stored in your namespace. There are five types of access to DNS names: read, write, delete, test, and control.

Chapter 2

Planning Your Namespace

Before you install DNS, you need to make the following decisions:

- Whether to create a single-directory or a hierarchical namespace.
- How to design your single-directory or hierarchical namespace. For example, how will you name the objects and the directories in your namespace?
- How many DNS servers you need in your network.
- Which nodes will be DNS servers and where in your network to locate them.
- How to distribute your namespace if you have more than one DNS server.

This chapter provides guidelines for making these decisions.

2.1 Deciding the Namespace Structure

Use the following criteria to decide whether to create a single-directory or a hierarchical namespace:

- A single-directory namespace is appropriate if your network consists of fewer than 100 nodes.

  If you create a single-directory namespace, you can have one DNS server in your network that stores the root directory or you can distribute the entire root directory to several DNS servers. See Section 2.4 for details.

- A hierarchical namespace is best if your network has more than 100 nodes.

  If you create a hierarchical namespace, you can have one DNS server in your network that stores all the directories in the hierarchy. Alternatively, you can distribute all or some of the directories to several DNS servers. See Section 2.4 for details.

For ease of use, you might want to begin with a single-directory namespace. As you become more familiar with DNS and as your network grows, you can restructure your single-directory namespace into a hierarchical namespace by creating additional directories. Section 2.1.2 provides guidelines for designing a hierarchical namespace; Chapter 3 tells you how to create one.

NOTE

> Digital recommends that DNS managers gain experience with DNS by first using a single-directory namespace.
>
> Switching to a hierarchical namespace requires DNS managers to re-define all DNS object names.

2.1.1 Designing a Single-Directory Namespace

Designing a single-directory namespace involves storing DNS object names in the root directory. Because DNS names must be unique within a directory, you may want to set up a networkwide policy for choosing DNS names to avoid conflicts. All DNS managers in the network should participate in setting this policy. For example, all names used by DFS could begin with DISK_.

NOTE

> If a single-directory namespace suits your needs, you can go directly to Section 2.2, which tells you how to determine the number of DNS servers for your network.

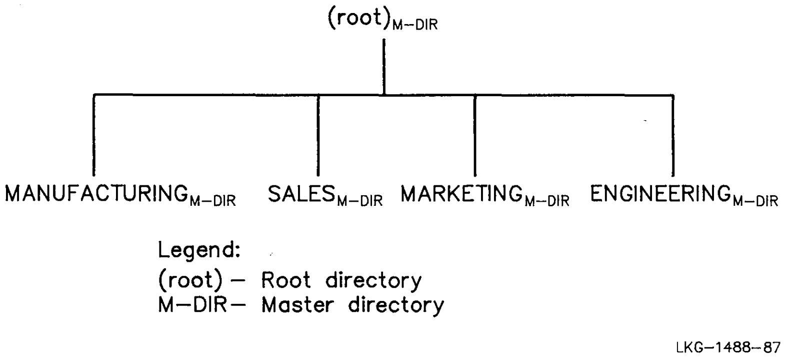
2.1.2 Designing a Hierarchical Namespace

When designing a hierarchical namespace, you must make a basic organizational decision: What kind of hierarchy will your namespace represent? It is recommended that the directories in your namespace represent the functional structure of your organization. All DNS managers in your network may want to adopt a policy for choosing directory names to avoid conflicts.

Figure 2-1 shows the top level of a hierarchical namespace that will be structured by functional groups within the organization. The top level of the hierarchy contains directories named after the functional areas, MANUFACTURING, SALES, MARKETING, and ENGINEERING. These directories will store object names for the applications used by those functional groups.

The directories shown in Figure 2-1 were created with the DNS$CONTROL CREATE DIRECTORY command. The CREATE DIRECTORY command creates **master directories** in the namespace. (See Chapter 3 for instructions on how to build a hierarchical namespace.)

Figure 2-1: The Top of the Namespace Hierarchy



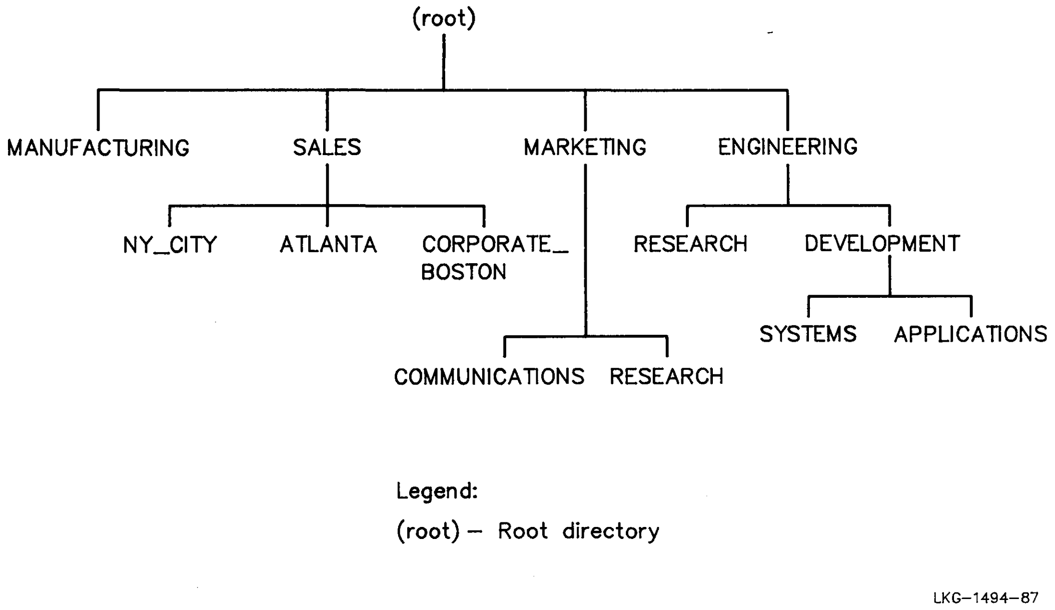
NOTE

> Take special care when you create the top level of a hierarchical namespace. You must specify directories or groupings that you know are stable and will not change. Instability at the top of the hierarchy can require frequent redesigning of major portions of the namespace.

When you design your namespace, decide on the top two or three levels. Figure 2-2, an extension of Figure 2-1, shows how a logical structure might begin to evolve as you build the namespace hierarchy.

In Figure 2-2, the directory ENGINEERING has two **child directories**, one named RESEARCH, the other named DEVELOPMENT. The DEVELOPMENT directory also has two **child directories**, one named SYSTEMS, the other named APPLICATIONS. Chapter 3, Section 3.4, contains the CREATE DIRECTORY commands that actually create the ENGINEERING portion of the namespace hierarchy shown in Figure 2-2.

Figure 2-2: Several Levels of a Namespace Hierarchy



Object names stored in directories at lower levels of the hierarchy represent network resources referenced by smaller user groups. It is recommended that the directories at the lowest level of the hierarchy support up to 100 nodes. However, you may want to support fewer than 100 nodes in order to separate groups of users according to their access control requirements. (See Chapter 3 for a discussion of access control.)

2.2 Deciding the Number of DNS Servers

While one DNS server can serve your entire network, use the following general rules to choose the number of DNS servers:

- Install one DNS server for a network of up to 100 nodes, two or more DNS servers for networks of more than 100 nodes.

  However, even in networks of up to 100 nodes, a second DNS server should be installed to increase the availability of DNS names.
- Install at least one DNS server for each Ethernet on which you have applications that use DNS.

Any node on which you install DNS must meet the hardware and software requirements specified in the *VAX Distributed Name Service Installation Guide.*

NOTE

> Because applications such as DFS and RSM V2.0 cannot run without DNS, you should consider having more than one DNS server in your network. This increases the likelihood that DNS will be available if a node or communications failure puts one of your DNS servers off line.

2.3 Deciding the Location of DNS Servers

To best locate a DNS server in your network, consider which nodes are most often available and provide rapid responses to requests. DNS server nodes should not be subject to frequent down time. Nor should they carry a heavy load of other work that would degrade DNS name lookup performance.

NOTE

> Because DNS updates should be able to run to completion, the DECnet connections between your DNS servers should be stable.

2.4 Distributing Your Namespace On Multiple DNS Servers

This section summarizes the general rules for distributing single-directory and hierarchical namespaces.

2.4.1 Distributing a Single-Directory Namespace

In a single-directory namespace, the DNS installation automatically creates a master copy of the root directory when you install your first DNS server. You can store object names in that root directory immediately after installation. Then when you install subsequent DNS servers, the installation procedure automatically copies a read-only copy of the entire root directory to the new clearinghouse you are creating. Therefore, in a single-directory namespace, you do not have to issue any commands to replicate the root directory on multiple DNS servers; the installation procedure does this for you.

There is no limit to the number of DNS servers you can have in your network, even if you have a single-directory namespace.

2.4.2 Distributing a Hierarchical Namespace

After you've installed your first DNS server and have created a hierarchy of DNS directories, you can distribute read-only copies of the master directories you created to any clearinghouse on any DNS server. You distribute directories by using the DNS$CONTROL COPY DIRECTORY command, as described in Chapter 3.

Note that when you create a clearinghouse during installation, the directory that you specify to contain the clearinghouse object is copied to the clearinghouse. Therefore, this is another means by which a directory is copied to a clearinghouse.

You should locate read-only directories on servers that are physically close to the applications that use the names stored in them. This improves the response time of DNS name lookup operations.

Also, distributing read-only directories on multiple DNS servers increases the likelihood that the DNS names stored in them are always available, even if one or more DNS servers becomes unavailable.

2.5 Installing DNS Software

Now that you are familiar with DNS concepts and namespace design, you should be ready to install DNS software. Refer to the *VAX Distributed Name Service Installation Guide* for instructions.

Chapter 3
Managing DNS

This chapter tells you how to use the DNS$CONTROL program to perform the following DNS management tasks:

- Manage access control
- Construct a hierarchical namespace
- Display information stored in the namespace
- Manage DNS servers
- Distribute the namespace
- Restructure the namespace

3.1 Using DNS$CONTROL

The VAX Distributed Name Service Control Program (DNS$CONTROL) is a management program that enables users to create and manage a DNS namespace. On-line help is available for all DNS$CONTROL commands (see Section 3.1.4).

Table 3-1 summarizes the DNS$CONTROL commands. The remainder of the chapter describes each DNS management task in detail. Chapter 4 details the format and function of DNS$CONTROL commands.

3.1.1 Requirements for Using DNS$CONTROL

DNS$CONTROL needs 11,000 bytes of BYTLM in order to run successfully. If DNS$CONTROL runs out of pool space, the error message SYSTEM-F-EXBYTLM may be issued. If you get this message, use the VMS AUTHORIZE utility to increase the value of the BYTLM quota for the account from which DNS$CONTROL was run.

3.1.2 Invoking DNS$CONTROL

Enter the following command to invoke DNS$CONTROL:

```
$ RUN SYS$SYSTEM:DNS$CONTROL
```

The DNS$CONTROL prompt (DNS>) will then be displayed as follows:

```
DNS>
```

You can enter any command listed in Table 3-1 at the DNS> prompt.

3.1.3 Exiting DNS$CONTROL

To exit DNS$CONTROL, enter EXIT or press CTRL/Z.

3.1.4 Getting Help on DNS$CONTROL

If you need information on any DNS$CONTROL command, enter the following command at the DNS> prompt:

HELP *command*

where *command* is any command listed in Table 3-1. If you enter a command plus a keyword, you will get information on the allowable parameters.

If you specify a command followed by an asterisk (*) in place of a keyword, you will get information on all allowable keywords for that command. For example, if you enter ADD *, information on all ADD commands will be displayed.

On-line Help also tells you whether additional information on specified parameters or command examples is available.

3.1.5 DNS$CONTROL Command Syntax

A DNS$CONTROL command consists of three parts:

1. **Command verb:** The first word of any command listed in Table 3-1. It expresses the action that the command performs.
2. **Keyword or keywords:** Designates the entity on which the command acts. An entity can be a namespace, a clearinghouse, a directory, and so on. See Table 3-1.
3. **Parameters:** These are specified in Chapter 4.

You can abbreviate any command verb, keyword, or parameter to its fewest number of unique letters.

3.2 Summary of DNS$CONTROL Commands

Table 3-1 summarizes the DNS$CONTROL commands. The commands appear along with their keywords. For example, DELETE is a command verb. CLEARINGHOUSE, DIRECTORY, GROUP, LINK, and OBJECT are its keywords.

Table 3-1: DNS$CONTROL Commands and Functions

Command	Function
ADD ACCESS	Provides access rights for individuals and groups.
ADD CLEARINGHOUSE	Adds an existing clearinghouse to the local DNS server.
ADD MEMBER	Adds an individual or group to an access control group.
COPY DIRECTORY	Copies a directory to a clearinghouse.
CREATE DIRECTORY	Creates a directory and places it in a specified clearinghouse.
CREATE GROUP	Creates and names an access control group.
CREATE LINK	Creates a link from one namespace entry to another.

(continued on next page)

Table 3-1 (Cont.): DNS$CONTROL Commands and Functions

Command	Function
DELETE CLEARINGHOUSE	Deletes a clearinghouse.
DELETE DIRECTORY	Deletes a directory.
DELETE GROUP	Deletes a group.
DELETE LINK	Deletes a link.
DELETE OBJECT	Deletes an object.
EXIT	Exits DNS$CONTROL.
HELP	Displays information on command formats and functions.
REBUILD CLEARINGHOUSE	Reclaims unused space in a specified clearinghouse.
REBUILD DIRECTORY	Performs two functions: 1. Assigns a new master directory. 2. Excludes a permanently unavailable read-only directory from a set of copies of that directory.
REMOVE ACCESS	Removes access rights from an individual or group entry.
REMOVE CLEARINGHOUSE	Removes a clearinghouse from a list of clearinghouses known to a particular DNS server. The clearinghouse can then be moved to a new node or disk.
REMOVE DIRECTORY	Removes a copy of a directory from a clearinghouse.
REMOVE MEMBER	Removes a member from a group.
SET CLEARINGHOUSE	Performs two functions: 1. Modifies the RMS buffer count for a clearinghouse. 2. Establishes the time for the next scheduled update to this clearinghouse.

(continued on next page)

Table 3-1 (Cont.): DNS$CONTROL Commands and Functions

Command	Function
SET DIRECTORY	Performs two functions: 1. Specifies whether a directory may contain clearinghouse objects. 2. Specifies how often the update function should try to update directory copies.
SET TIMEZONE	Modifies the value of an offset between Greenwich Mean Time and the time in the node's timezone.
SHOW ACCESS	Displays access rights for clearinghouses, directories, groups, links, or objects.
SHOW ACTIVE CLEARINGHOUSES	Displays information about clearinghouses currently enabled on this node.
SHOW CHARACTERISTICS	Displays values of certain attributes for child pointers, clearinghouses, directories, links, and objects.
SHOW CHILD	Displays the child pointers of a parent directory.
SHOW CLEARINGHOUSE	Displays information about a clearinghouse, such as number of directories it contains, its RMS buffer count, its counters, and so on.
SHOW DIRECTORY	Displays attributes and a list of objects, child pointers, or link information for a directory.
SHOW GROUP	Displays attribute and member information for a specified group.
SHOW KNOWN CLEARINGHOUSES	Displays information for all clearinghouses known to the local DNS server.
SHOW LINK	Displays names of all attributes or the value of a specific attribute for a link.

(continued on next page)

Table 3-1 (Cont.): DNS$CONTROL Commands and Functions

Command	Function
SHOW NAMESERVER	Displays namespace name, unique identifier (UID), timezone information, and counters for a DNS server. The UID consists of network address and time information that is assigned by DNS to the namespace and to each entry in it.
SHOW OBJECT	Displays all attribute names or the value of a specific attribute for a specified object.
SHOW VERSION	Displays version number of the installed DNS software.
START CLEARINGHOUSE	Starts a specified clearinghouse and zeroes all counters associated with it.
STOP CLEARINGHOUSE	Stops the activity of a clearinghouse.
UPDATE DIRECTORY	Updates a directory.

3.3 Managing Access Control

You can protect or make available information in the namespace by granting or denying users access to that information. You can also specify how access control rights propagate to object entries created in a directory.

DNS installation automatically creates access control entries that allow all users to read, write, delete, test and control all namespace entries. After installation, you can modify access rights as necessary.

Access control values for an object are stored in the object's DNS$ACS attribute. (See Appendix B for a summary of all DNS attributes.) Access control values define who can access an object entry and what operations they can perform.

3.3.1 Displaying Access Rights

To determine who has access to a namespace entry, issue the DNS$CONTROL SHOW ACCESS command. For example, the following command displays the names of all users who have access to the object ENGINEERING.DEVELOPMENT.SALES_FORCE.

```
DNS> SHOW ACCESS OBJECT ENGINEERING.DEVELOPMENT.SALES_FORCE RET

    Entry______________APPLE::SYSTEM _ (BANANA_NS:.DNS$IV.APPLE.SYSTEM)
      Rights___________READ WRITE DELETE TEST CONTROL

    Entry______________*::* _ (BANANA_NS:.DNS$IV.*.*)
      Rights___________READ TEST

    Entry______________BANANA_NS:.ENGINEERING.DEVELOPMENT.ACL
      Flags____________GROUP
      Rights___________READ WRITE DELETE TEST CONTROL
```

3.3.2 Users and Access Control

In DNS, a user identification is expressed as:

node-name::*user-name*

where

node-name	is the name of the node from which the request is coming
user-name	is the account that the user is logged into on that node.

For access control purposes, you can specify users in three ways:

1. As individuals
2. As implicit groups
3. As explicit groups

An **individual** identifies one and only one user, such as MYNODE::SMITH. An **implicit group** identifies a collection of users through the use of wildcards (see the Glossary for a definition of wildcard). For example, the access control entry containing MYNODE::* grants access to all users on node MYNODE. An **explicit group** also identifies a collection of users, but the group is a DNS object. This object contains a list of members who are treated as a single entity for the purpose of access control.

3.3.3 Creating and Managing Groups

When you have a group of people with common access rights, you can treat them as a unit if you create a **group** in which all are **members**.

You create a group by issuing the DNS$CONTROL CREATE GROUP command as shown here:

CREATE GROUP *grp-name*

The group's unique identifier (UID) is displayed on your screen after you issue the command, as in the following example:

```
DNS> CREATE GROUP HOME_OFFICE.PERSONNEL [RET]
UID____________AA-00-04-00-8e-09-19-98-26-45-72-36-d2-f1
DNS>
```

To show all the groups in a directory, issue the DNS$CONTROL SHOW DIRECTORY command as shown here:

SHOW DIRECTORY *dir-name* KNOWN GROUPS

3.3.3.1 Adding and Removing Group Members

A group object contains a list of members in the attribute DNS$Members.

To add an individual or implicit group to an explicit group, issue the DNS$CONTROL ADD MEMBER command as shown here:

ADD MEMBER *mem-name* GROUP *grp-name*

For example:

```
DNS> ADD MEMBER GANWAY::SPARTAN GROUP ENGINEERING.DEVELOPMENT.ACL [RET]
```

To add a group to another group, issue the ADD MEMBER command as shown here:

ADD MEMBER *mem-name* GROUP *grp-name* /FLAGS=(GROUP)

For example:

```
DNS> ADD MEMBER ENGINEERING.DEVELOPMENT.ACL GROUP HOME.OFFICE.PERSONNEL
/FLAGS=(GROUP)
```

This command makes the group, ENGINEERING.DEVELOPMENT.ACL a member of the group, HOME.OFFICE.PERSONNEL.

Issue the REMOVE MEMBER command to remove a member from a group. See Chapter 4 for more information on command function and format.

3.3.3.2 Displaying Group Members

To display the members of a group, enter a command such as the following:

```
DNS> SHOW GROUP ENGINEERING.DEVELOPMENT.ACL KNOWN MEMBERS RET
```

This command displays information such as the following on your screen:

```
Member____________GANWAY::SPARTAN (BANANA_NS:.DNS$IV.GANWAY.SPARTAN)
Member____________GANWAY::SYSTEM (BANANA_NS:.DNS$IV.GANWAY.SYSTEM)
Member____________GANWAY::DNS$SERVER (BANANA_NS:.DNS$IV.GANWAY.DNS$SERVER)
```

You could also issue the DNS$CONTROL SHOW GROUP command to determine whether a member is a part of a specified group. The format for the command is:

SHOW GROUP *grp-name* MEMBER *mem-name*

The information returned tells you whether *mem-name* is a member of group *grp-name*. Members of a group can be individuals, implicit groups, or explicit groups.

3.3.4 Types of Access Rights

There are five types of access that you can grant to a user: read, write, delete, test, and control. Table 3-2 describes these access rights.

Table 3-2: DNS Access Rights

Entry	Right	Access Rights Conveyed
OBJECT, GROUP, LINK	Read	May examine any attribute.
	Write	May change any attribute of an entry except for the access control set attribute.
	Delete	May delete the entry or an attribute of the entry.
	Test	May test value of an attribute.
	Control	May alter the access control set attribute.
CLEARINGHOUSE	Read	May read any attribute of the clearinghouse.
	Write	May add or remove copies. May change a read-only directory to a master directory or vice versa. May alter attributes of the clearinghouse.
	Delete	May delete a clearinghouse.
	Test	May test the value of an attribute.
	Control	May alter the access control set. May move a clearinghouse to another DNS server.
DIRECTORY	Read	May examine any directory attribute. May examine the names of entries in a directory.
	Write	May create object, directory, and link entries.
	Delete	May delete a directory.
	Test	May test the value of an attribute.
	Control	May alter the access control set. May alter the replica list.

3.3.5 Access Flags for the ADD ACCESS Command

You use the DNS$CONTROL ADD ACCESS command to modify user access rights. You can specify three flags with the ADD ACCESS command: GROUP, NOPROPAGATE, and DEFAULT.

1. **GROUP:** Indicates that the member is a DNS group object. If you do not specify this flag, DNS regards the member as an individual or an implicit group.
2. **NOPROPAGATE:** Used for controlling access to directories. Indicates that the access rights granted should not be automatically copied to child directories of this directory. If you do not specify the NOPROPAGATE flag, the access rights for the directory propagate to all new child directories. This flag does not affect access rights to existing child directories.
3. **DEFAULT:** Used for adding access to the contents of directories. Indicates that the access rights granted should be automatically copied to new objects, groups, and links created in the directory. This allows the access rights to be applied automatically to new objects, groups, and links in the directory, without the DNS manager explicitly having to grant the access rights. This flag does not affect access rights to existing objects, groups, or links.

NOTE

> When the DEFAULT flag is used, the access rights granted do not apply to the directory itself. (Access to the directory is neither granted nor denied when the DEFAULT flag is used.) Rather, the access rights apply to newly created objects, groups, or links in the directory.

3.3.6 Adding Access Rights

Use the DNS$CONTROL ADD ACCESS command to grant users access to directories, clearinghouses, groups, objects, and links.

You can grant an individual or group access rights in two ways:

1. With the DEFAULT flag set
2. Without the DEFAULT flag

If you specify the DEFAULT flag with the ADD ACCESS command, the access rights that you grant in the command apply to newly created objects, groups, or links in the directory. If you do not specify the DEFAULT flag, the access rights you grant in the ADD ACCESS command apply to the directory itself but not to the entries in the directory.

Example of adding access rights with the DEFAULT flag:

```
DNS> ADD ACCESS MYNODE::SMITH DIRECTORY ENGINEERING.RESEARCH
     /FLAGS=(DEFAULT) /RIGHTS=(READ) [RET]
```

This command grants the individual MYNODE::SMITH read access to all new objects, groups, and links that will be created in the directory ENGINEERING.RESEARCH. The command does not grant MYNODE::SMITH any access rights to the directory ENGINEERING.RESEARCH. It also does not affect access rights to existing objects, groups, or links in the directory.

Example of adding and modifying access rights without the DEFAULT flag:

```
DNS> ADD ACCESS MYNODE::SMITH DIRECTORY ENGINEERING.RESEARCH
     /RIGHTS=(READ,WRITE) [RET]

DNS> ADD ACCESS MYNODE::SMITH DIRECTORY ENGINEERING.RESEARCH
     /RIGHTS=(DELETE) [RET]
```

These two commands grant the individual MYNODE::SMITH delete access to directory ENGINEERING.RESEARCH. Note that the access rights granted in the second command supersede the access rights granted in the first command.

3.3.7 Removing Access

Use the DNS$CONTROL REMOVE ACCESS command to delete access rights that you have granted to users. If you have granted a user a set of access rights, such as read, write, and delete access, you cannot remove individual access rights in that set; you must remove the entire set of access rights.

NOTE

> If you specified the DEFAULT flag when you added access rights, you cannot remove those access rights unless you also specify the DEFAULT flag in the REMOVE ACCESS command.

Example of removing access rights without the DEFAULT flag:

```
DNS> ADD ACCESS MYNODE::SMITH DIRECTORY ENGINEERING.RESEARCH
     /RIGHTS=(READ,WRITE) RET

DNS> REMOVE ACCESS MYNODE::SMITH DIRECTORY ENGINEERING.RESEARCH RET
```

In this example, the first command grants the individual MYNODE::SMITH read and write access to the directory ENGINEERING.RESEARCH. The second command removes the access granted by the first command.

3.3.8 Determining Access

When you grant or deny access to an entry, DNS ignores all access control entries that have the DEFAULT flag set. Then it looks for an access control entry for the user who is requesting access. If an entry exists for the individual, DNS grants or denies access, depending on the rights field in the access control entry for the individual.

If DNS does not find an entry for the individual, it searches for implicit and explicit group access control entries to which the user might belong. DNS searches for an entry until it finds one that grants the user the required access or until there are no more entries. In the following example, user MYNODE::SMITH is denied read access to the directory ENGINEERING.DEVELOPMENT and user MYNODE::SOMEBODY is granted read access to the same directory.

```
DNS> ADD ACCESS MYNODE::SMITH DIRECTORY ENGINEERING.DEVELOPMENT
     /RIGHTS=(WRITE,DELETE) RET

DNS> ADD ACCESS MYNODE::* DIRECTORY ENGINEERING.DEVELOPMENT
     /RIGHTS=(READ) RET
```

3.3.9 Required Access Rights for DNS Servers

A DNS server must always run in the account, [DNS$SERVER]. The [DNS$SERVER] account requires an access control entry.

The [DNS$SERVER] account needs read, write, delete, and test access to the clearinghouse object and clearinghouse entry for the clearinghouse local to the server.

3.3.10 Access Rights Summary

Initially, access rights are set by the DNS installation procedure. Upon successful installation of DNS, all users have read, write, delete, control, and test access.

To manage access to entries in the namespace, you use the DNS$CONTROL ADD ACCESS and REMOVE ACCESS commands. These commands control access in the following ways:

- They add an access control entry to or remove it from the DNS$ACS attribute of a clearinghouse, directory, object, or link.
- They specify how to propagate those rights for child directories of the directory specified in the command.
- They specify how to propagate those rights for objects, groups, and links in the specified directory.

To display access control entries, issue the DNS$CONTROL SHOW ACCESS command, using the keyword, CLEARINGHOUSE, DIRECTORY, OBJECT, GROUP, or LINK.

3.4 Constructing a Hierarchical Namespace

A hierarchical namespace consists of a hierarchy of DNS directories under the root directory. The root directory is created when DNS is installed. You use the DNS$CONTROL CREATE DIRECTORY command to build other directories under the root. Any directory you create with this command is a master directory. You create read-only copies of a master directory with the DNS$CONTROL COPY DIRECTORY command, as discussed later in this chapter.

You can create a hierarchical structure in your namespace whether one or several DNS servers are installed in your network. If you have only one DNS server, all the directories will be located in a single clearinghouse. If you have more than one DNS server, the hierarchy of directories will be located in multiple clearinghouses.

The following sections present two examples of hierarchical namespaces. The first example shows how to create a hierarchical namespace in a network that has one DNS server; the second shows how to create a hierarchical namespace in a network that has more than one DNS server.

3.4.1 Creating a Hierarchical Namespace in a Single-Server Network

Having one DNS server in your network means that you have one clearinghouse on that server. Any DNS directory that you create will be located in that clearinghouse.

Assume that the name of the clearinghouse is MYNODE_CH. To create a directory called ENGINEERING and place it in that clearinghouse, enter the following command:

```
DNS> CREATE DIRECTORY ENGINEERING [RET]
```

You can then create the child directories specified in following commands:

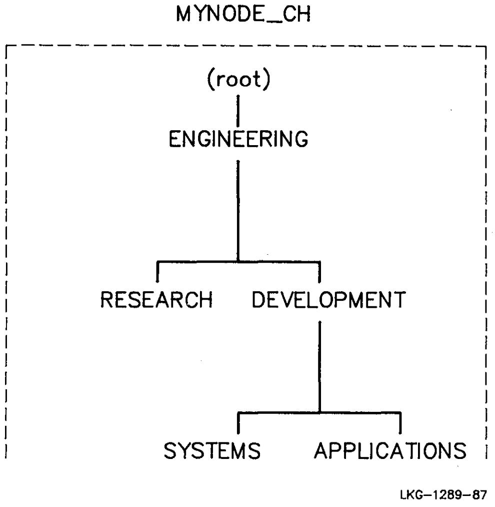
```
DNS> CREATE DIRECTORY ENGINEERING.RESEARCH [RET]
DNS> CREATE DIRECTORY ENGINEERING.DEVELOPMENT [RET]
```

The child directories reside in MYNODE_CH under directory ENGINEERING, as shown in Figure 3-1. By default, the child directories are placed in clearinghouse MYNODE_CH because DNS automatically places a child directory in the same clearinghouse as the master copy of its parent directory. This is why you did not have to specify which clearinghouse would contain the directories ENGINEERING.DEVELOPMENT and ENGINEERING.RESEARCH when you created the directories.

Figure 3-1 also shows directories SYSTEMS and APPLICATIONS, which were then created and placed in directory ENGINEERING.DEVELOPMENT by the following two commands:

```
DNS> CREATE DIRECTORY ENGINEERING.DEVELOPMENT.SYSTEMS [RET]
DNS> CREATE DIRECTORY ENGINEERING.DEVELOPMENT.APPLICATIONS [RET]
```

Figure 3-1: Hierarchy of Directories in a Single-Server Network

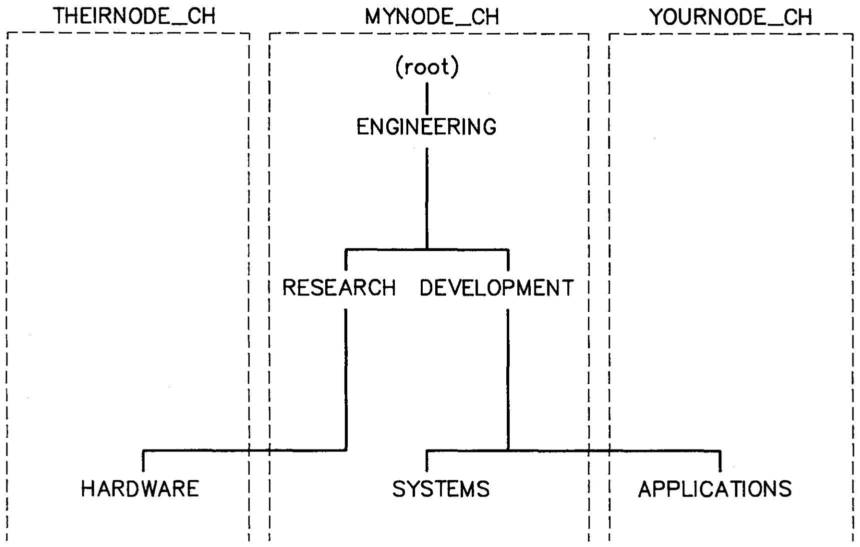


3.4.2 Creating a Hierarchical Namespace in a Multiserver Network

Having more than one DNS server in your network means that you have multiple clearinghouses, that is, one clearinghouse on each server. Any directory that you create will be located in the clearinghouse that you specify with the CREATE DIRECTORY command. For example, if you have three clearinghouses (MYNODE_CH, YOURNODE_CH, and THEIRNODE_CH), you would enter the following commands to set up the hierarchy shown in Figure 3-2. Note that UIDs are returned for each CREATE DIRECTORY command.

```
DNS> CREATE DIRECTORY ENGINEERING CLEARINGHOUSE MYNODE_CH [RET]
UID_____________________________aa-00-04-00-11-40-2d-87-3d-b4-6f-90-00
DNS> CREATE DIRECTORY ENGINEERING.RESEARCH [RET]
UID_____________________________aa-00-04-00-09-37-2d-87-6d-e5-6f-90-00
DNS> CREATE DIRECTORY ENGINEERING.RESEARCH.HARDWARE CLEARINGHOUSE
     THEIRNODE_CH [RET]
UID_____________________________aa-00-04-00-6e-24-2d-87-19-e4-6f-90-01
DNS> CREATE DIRECTORY ENGINEERING.DEVELOPMENT [RET]
UID_____________________________aa-00-04-00-11-04-3d-78-19-34-6f-90-01
DNS> CREATE DIRECTORY ENGINEERING.DEVELOPMENT.SYSTEMS [RET]
UID_____________________________aa-00-04-00-6e-37-2d-87-4d-01-90-6f-00
DNS> CREATE DIRECTORY ENGINEERING.DEVELOPMENT.APPLICATIONS CLEARINGHOUSE
     YOURNODE_CH [RET]
UID_____________________________aa-00-04-00-10-35-19-34-2d-78-09-22-11
```

Figure 3-2: Hierarchy of Directories in a Multiserver Network



Note that you did not have to specify which clearinghouse would contain the directories ENGINEERING.RESEARCH, ENGINEERING.DEVELOPMENT, and

ENGINEERING.DEVELOPMENT.SYSTEMS. This is because DNS automatically places a child directory in the same clearinghouse as the master copy of its parent directory.

3.5 Displaying Information Stored in the Namespace

Certain procedures require you to gather information about entries in the namespace or about the namespace itself. The DNS$CONTROL SHOW commands enable you to display the information that is stored in your namespace.

The namespace contains four types of information:

1. Access control information
2. Attribute information detailing the characteristics of an entry
3. Contents of:
   - Clearinghouses in the namespace
   - Directories in the namespace
   - Objects, links, and child pointers in each directory
   - Members in each group
4. Statistical information that may help you evaluate DNS performance.

The following sections discuss these types of information and tell how to use the SHOW commands to obtain each.

3.5.1 Displaying Access Control Information

The DNS$CONTROL SHOW ACCESS command displays the users that have access rights to clearinghouses, directories, objects, links, and groups. It also displays the access rights the users have and any flags that apply.

The following examples show information returned by various SHOW ACCESS commands. In the examples, the information in parentheses represents the DNS internal form of the entry.

Example 1: Showing access to clearinghouses

```
DNS> SHOW ACCESS CLEARINGHOUSE BANANA_CH [RET]

Entry__________BANANA::SPEARS _  (BANANA_NS:.DNS$IV.BANANA.SPEARS)
 Rights________READ WRITE DELETE TEST CONTROL

Entry__________BANANA::DNS$SERVER _ (BANANA_NS:.DNS$IV.BANANA.DNS$SERVER)
 Rights________READ WRITE DELETE TEST CONTROL

Entry__________*::* _ (BANANA_NS:.DNS$IV.*::*)
 Rights________READ WRITE DELETE TEST CONTROL

Entry__________BANANA::system _ (BANANA_NS:.DNS$IV.BANANA.SYSTEM)
 Rights________READ WRITE DELETE TEST CONTROL
```

Example 2: Showing access to directories

```
DNS> SHOW ACCESS DIRECTORY ENGINEERING.DEVELOPMENT [RET]

Entry__________BANANA::SPEARS _ (BANANA_NS:.DNS$IV.BANANA.SPEARS)
 Flags_________NOPROPAGATE
 Rights________READ WRITE DELETE TEST CONTROL

Entry__________BANANA::DNS$SERVER _ (BANANA_NS:.DNS$IV.BANANA.DNS$SERVER)
 Flags_________NOPROPAGATE
 Rights________READ WRITE DELETE TEST CONTROL

Entry__________BANANA_NS:.BANANA_CH
 Flags_________NOPROPAGATE
 Rights________READ WRITE DELETE TEST CONTROL

Entry__________*::* _ (BANANA_NS:.DNS$IV.*.*)
 Flags_________DEFAULT PROPAGATE
 Rights________READ TEST

Entry__________*::* _ (BANANA_NS:.DNS$IV.*.*)
 Flags_________PROPAGATE
 Rights________READ TEST

Entry__________BANANA_NS:.ENGINEERING.DEVELOPMENT.ACL
 Flags_________PROPAGATE GROUP
 Rights________READ WRITE DELETE TEST CONTROL

Entry__________BANANA_NS:.ENGINEERING.MSS.ACL
 Flags_________DEFAULT PROPAGATE GROUP
 Rights________READ WRITE DELETE TEST CONTROL
```

Example 3: Showing access to a clearinghouse object

```
DNS> SHOW ACCESS OBJECT ENGINEERING.DEVELOPMENT.GANWAY_CH RET

Entry__________GANWAY::SPARTAN _ (BANANA_NS:.DNS$IV.GANWAY.SPARTAN)
 Rights________READ WRITE DELETE TEST CONTROL

Entry__________... _ (BANANA_NS:.DNS$IV....)
 Rights________READ TEST

Entry__________*.* _ (BANANA_NS:.DNS$IV.*.*)
 Rights________READ TEST

Entry__________BANANA_NS:.ENGINEERING.DEVELOPMENT.ACL
 Flags_________GROUP
 Rights________READ WRITE DELETE TEST CONTROL
```

Example 4: Showing access to an object

```
DNS> SHOW ACCESS OBJECT ENGINEERING.DEVELOPMENT.SALES_FORCE RET

Entry__________BANANA::SPEARS _ (BANANA_NS:.DNS$IV.BANANA.SPEARS)
 Rights________READ WRITE DELETE TEST CONTROL

Entry__________*::* _ (BANANA_NS:.DNS$IV.*.*)
 Rights________READ TEST

Entry__________BANANA_NS:.ENGINEERING.DEVELOPMENT.ACL
 Flags_________GROUP
 Rights________READ WRITE DELETE TEST CONTROL
```

Example 5: Showing access to a link

```
DNS> SHOW ACCESS LINK ENG_SALES RET

Entry__________BANANA::SPEARS _ (BANANA_NS:.DNS$IV.BANANA.SPEARS)
 Rights________READ WRITE DELETE TEST CONTROL

Entry__________*::* _ (BANANA_NS:.DNS$IV.*.*)
 Rights________READ TEST
```

Example 6: Showing access to a group

```
DNS> SHOW ACCESS GROUP ENGINEERING.DEVELOPMENT.ACL RET

Entry__________BANANA::SPEARS _ (BANANA_NS:.DNS$IV.BANANA.SPEARS)
 Rights________READ TEST

Entry__________*::* _ (BANANA_NS:.DNS$IV.*.*)
 Rights________READ TEST
```

3.5.2 Displaying Attribute Information

Attributes are characteristics of directories, clearinghouses, links, objects, and child pointers. DNS defines a list of attributes for each of these entry types, as listed in Appendix B. Attributes and their values can be modified by DNS$CONTROL commands, by DNS server operation, and by applications that use DNS. Applications may add application-specific attributes to object entries created by the application.

There are two types of attributes: single-value and sets. Single-value attributes have one and only one value; sets have zero or more values.

The following sections describe why you might need the information contained in DNS attributes and how to obtain it.

3.5.2.1 Displaying Attribute Values

You can display the following two types of attribute information:

1. The value of a specified attribute for a specified clearinghouse, directory, object, link, or child pointer
2. The names of all known attributes of a specified clearinghouse, directory, object, link, or child pointer

Table 3-3 tells you what DNS$CONTROL command to use to obtain the information contained in DNS attributes.

Table 3-3: Accessing Information Stored in DNS Attributes

If you want to know...	Enter this command
Who has access to a specified clearinghouse, directory, or object	SHOW ACCESS
The name of the application to which an object belongs	SHOW CHARACTERISTICS OBJECT
The version number of the application software under which an object was created	
Where in the network an object resides	
The following information for a directory: • Location of each copy (node name/node address and clearinghouse) • Type of each directory (master or read-only) • Convergence value of each (HIGH or LOW) • Whether directory can store a clearinghouse object (ALLOWED or DISALLOWED)	SHOW CHARACTERISTICS DIRECTORY
Timestamp information for determining whether directories are similarly updated	
Current update timing information to determine whether the attribute value should be reset	
Whether a directory is allowed to store clearinghouse objects	
The following information for a child directory: • Location of each copy (node name/node address and clearinghouse) • Type of each directory (master or read-only)	SHOW CHARACTERISTICS CHILD
The name of the entry to which a link points and its expiration and extension time	SHOW CHARACTERISTICS LINK
The content of a clearinghouse	SHOW CHARACTERISTICS CLEARINGHOUSE
The names of members in a group	SHOW CHARACTERISTICS GROUP

DNS provides two techniques for retrieving attribute values from the namespace.

1. Use the DNS$CONTROL SHOW CHARACTERISTICS command for a specific clearinghouse, directory, object, link, group, or child pointer.
2. Show the value of any single attribute for a specific clearinghouse, directory, object, link, group, or child pointer.

Displaying Characteristics

The DNS$CONTROL SHOW CHARACTERISTICS CLEARINGHOUSE command displays the directory copies stored in the clearinghouse.

The following is a display returned by the SHOW CHARACTERISTICS CLEARINGHOUSE command:

```
DNS> SHOW CHARACTERISTICS CLEARINGHOUSE MYNODE_CH [RET]

Directory_____________________BANANA_NS:.ENGINEERING
Directory_____________________BANANA_NS:.ENGINEERING.RESEARCH
Directory_____________________BANANA_NS:.ENGINEERING.DEVELOPMENT
Directory_____________________BANANA_NS:.ENGINEERING.DEVELOPMENT.SYSTEMS
Directory_____________________BANANA_NS:.
```

The DNS$CONTROL SHOW CHARACTERISTICS DIRECTORY command displays the following information for the specified directory:

- Whether the directory has been allowed to store clearinghouse objects
- Convergence value of the directory
- Date and time of the last successful update to the directory
- Address, replica type, and clearinghouse name for the directory

The following is a display returned by the SHOW CHARACTERISTICS DIRECTORY command:

```
DNS> SHOW CHARACTERISTICS DIRECTORY ENGINEERING.DEVELOPMENT [RET]

Clearinghouses_______________ALLOWED
Convergence__________________HIGH
Last Successful Update_______14-AUG-1987 13:36:56.30

Directory copies:

    Address___________________MYNODE (4.828)
    Replica Type______________Master
    Clearinghouse_____________BANANA_NS:.MYNODE_CH
    Address__________________HALVA (4.274)
    Replica Type_____________Read-only
    Clearinghouse____________BANANA_NS:.YOURNODE_CH
```

The DNS$CONTROL SHOW CHARACTERISTICS OBJECT command displays the values of the object's address, class, and version.

The following is a display returned by the SHOW CHARACTERISTICS OBJECT command:

```
DNS> SHOW CHARACTERISTICS OBJECT MYNODE_CH RET

Address______________________MYNODE (4.828)
Class________________________DNS$CLEARINGHOUSE
Version______________________V1.0
```

The DNS$CONTROL SHOW CHARACTERISTICS LINK command displays the name of the entry that the link points to, along with the expiration and extension times for the link.

The following is a display returned by the SHOW CHARACTERISTICS LINK command:

```
DNS> SHOW CHARACTERISTICS LINK ENG_SALES RET

Target___________________BANANA_NS:.ENGINEERING.DEVELOPMENT.SALES_FORCE
Expiration_______________Infinite
Extension________________None
```

The DNS$CONTROL SHOW CHARACTERISTICS CHILD command displays the address, replica type, and clearinghouse name for each copy of the child directory:

The following is a display returned by the SHOW CHARACTERISTICS CHILD command:

```
DNS> SHOW CHARACTERISTICS CHILD ENGINEERING.DEVELOPMENT RET

Address______________________MYNODE (4.828)
Replica Type_________________Master
Clearinghouse________________BANANA_NS:.MYNODE_CH

Address______________________HALVA (4.274)
Replica Type_________________Read-only
Clearinghouse________________BANANA_NS:.YOURNODE_CH
```

Displaying Single-Value Attributes

Use the DNS$CONTROL SHOW ATTRIBUTE command to retrieve the value of an attribute for a clearinghouse, directory, object, group, link, or child pointer. The value associated with the specified attribute is returned as follows:

```
DNS> SHOW DIRECTORY ENGINEERING.DEVELOPMENT ATTRIBUTE DNS$CONVERGENCE RET

Convergence___________________________HIGH
  Timestamp______14-AUG-1987 12:54:32.15 AA-00-04-00-8E-11
```

3.5.2.2 Displaying the Names of Known Attributes

Use the DNS$CONTROL SHOW KNOWN ATTRIBUTES command to retrieve the names of all known attributes for a specific child pointer, clearinghouse, directory, group, object, or link. The known attributes are returned as follows:

```
DNS>SHOW CLEARINGHOUSE BANANA_CH KNOWN ATTRIBUTES RET

 Attribute (Set) ______ DNS$ACS
 Attribute (Set) ______ DNS$CHDirectories
 Attribute (Single) ___ DNS$CHLastAddress
 Attribute (Set) ______ DNS$CHName
 Attribute (Single) ___ DNS$CHState
 Attribute (Single) ___ DNS$CHUID
 Attribute (Set) ______ DNS$CHUpPointers
 Attribute (Single) ___ DNS$NSNickname
 Attribute (Single) ___ DNS$NSUID
 Attribute (Single) ___ DNS$UID
 Attribute (Single) ___ DNS$UTS
```

3.5.3 Displaying the Contents of Namespace Entries

This section describes the SHOW commands that display information on the content of clearinghouses, directories, and groups.

3.5.3.1 Displaying the Contents of a Clearinghouse

The DNS$CONTROL SHOW ACTIVE CLEARINGHOUSES and SHOW KNOWN CLEARINGHOUSES commands display, along with other information, the number of directories in the specified clearinghouse. To obtain further information on the contents of clearinghouses, you can display attribute and access information, as described in Section 3.6.5.

3.5.3.2 Displaying the Contents of a Directory

The DNS$CONTROL SHOW DIRECTORY command displays the names of child pointers, objects, and links in a specified directory. The following sections describe how to display this information for a directory.

Displaying the Child Directories in a Directory

To find the names of all the directories in the namespace, you would work your way down the namespace hierarchy, starting at the root directory, with the following command.

The dot (.) in the command line specifies the root directory. This command displays the names of all directories stored in the root.

```
DNS> SHOW DIRECTORY . KNOWN CHILDREN [RET]

Child ___________________ENGINEERING
Child ___________________MARKETING
Child ___________________SALES
```

This information tells you that there are three directories in the root directory: ENGINEERING, MARKETING, and SALES.

You can continue to gather information on the child directories by issuing the SHOW DIRECTORY command for each of the directories, as follows:

```
DNS> SHOW DIRECTORY ENGINEERING KNOWN CHILDREN [RET]
DNS> SHOW DIRECTORY MARKETING KNOWN CHILDREN [RET]
DNS> SHOW DIRECTORY SALES KNOWN CHILDREN [RET]
```

Using the SHOW DIRECTORY command, you can follow each branch of the hierarchy until you have displayed all of the directories in your namespace hierarchy.

Displaying the Objects in a Directory

You can also use the SHOW DIRECTORY command to find the names of all objects in a directory, as in the following example:

```
DNS> SHOW DIRECTORY ENGINEERING.RESEARCH KNOWN OBJECTS [RET]

Object ___________ DEVELOPMENT_DISK
Object ___________ SOURCE_DISK
```

This display tells you that there are two objects in directory ENGINEERING.RESEARCH; one is named DEVELOPMENT_DISK, the other is named SOURCE_DISK.

Displaying the Links in a Directory

Use the SHOW DIRECTORY command in the following form to find the names of all links in a specified directory:

```
DNS> SHOW DIRECTORY ENGINEERING.RESEARCH KNOWN LINKS RET

LINK____________________LINKA
```

This display tells you that directory ENGINEERING.RESEARCH contains one link called LINKA.

3.5.3.3 Displaying the Contents of a Group

The DNS$CONTROL SHOW GROUP command displays:

- The value of an attribute
- The names of the attributes for the group
- The names of the members of the group
- Whether a member name is a member of the group

The following is a display returned by the SHOW GROUP command:

```
DNS> SHOW GROUP ENGINEERING KNOWN MEMBERS RET

Member_________________MYNODE::SPARTAN (BANANA_NS:.DNS$IV.MYNODE.SPEARS)
Member_________________HALVA::SPEARS (BANANA_NS:.DNS$IV.HALVA.SPEARS)
```

3.5.4 Displaying Statistical Information

DNS maintains statistical information on clearinghouses and on the namespace itself. See Section 3.6.5 for details on the statistical information you can display.

3.6 Managing DNS Servers

You will need to perform the following tasks to manage DNS servers:

- Start and stop clearinghouses
- Start and stop DNS servers
- Reclaim unused clearinghouse space
- Modify clearinghouse buffer counts
- Display clearinghouse information
- Set the DNS timezone
- Use a backup copy of the database file
- Maintain and tune the server

The following sections tell you how to perform these tasks.

3.6.1 Starting and Stopping Clearinghouses

The DNS$CONTROL START CLEARINGHOUSE command starts a clearinghouse and zeroes all counters associated with it.

The DNS$CONTROL STOP CLEARINGHOUSE command stops the activity of a clearinghouse. All directories in the clearinghouse become unavailable. Any operation in progress that uses those directories completes before the clearinghouse is stopped.

3.6.2 Starting and Stopping DNS Servers

Invoke one of the following command files to start or stop the DNS server:

- SYS$MANAGER:DNS$STARTUP: Starts DNS server and client software. When you start a DNS server, all of its counters are set to zero.
- SYS$MANAGER:DNS$STOP: Stops DNS server software only; it does not stop the client software. To stop the client software you must invoke the DNS$CLIENT_STOP command file.

3.6.3 Reclaiming Unused Clearinghouse Space

The clearinghouse database is stored in a Record Management System (RMS) keyed ISAM file. As you add and delete directories and objects in this file, you create unused disk space. To reclaim this unused space, issue the DNS$CONTROL REBUILD CLEARINGHOUSE command. This command invokes the RMS convert procedure.

3.6.4 Modifying Clearinghouse RMS Buffer Counts

The DNS$CONTROL SET CLEARINGHOUSE command enables you to modify the RMS buffer count of a clearinghouse. The RMS buffers make frequently used information more accessible. See Section 3.6.8 for more information on how to optimize the RMS buffer counts.

3.6.5 Displaying Clearinghouse Information

The following DNS$CONTROL commands enable you to display clearinghouse information:

- SHOW ACTIVE CLEARINGHOUSES
- SHOW KNOWN CLEARINGHOUSES
- SHOW CLEARINGHOUSE

SHOW ACTIVE CLEARINGHOUSES and **SHOW KNOWN CLEARINGHOUSES**:

These commands display the following information for clearinghouses that are currently active or known:

- Clearinghouse name: The name you specified during installation.
- Clearinghouse state: ON/OFF/INITIALIZING.
- Clearinghouse file name: The file specification for the clearinghouse file.
- RMS buffer count: The number of RMS buffers used for operation of the clearinghouse.
- Directories in this clearinghouse: The number of directories in this clearinghouse.

- UID: The unique identifier assigned to the clearinghouse when it was created.

The following is a sample display generated by the SHOW ACTIVE CLEARINGHOUSES command:

```
Clearinghouse name ________________ HOME_OFFICE
Clearinghouse state _______________ On
Clearinghouse filename ____________ sys$sysroot:[dns$server]HOME_OFFICE.dns
RMS buffer count __________________ 3
Directories in this clearinghouse _ 2
UID _________________________aa-00-04-00-8e-11-80-e0-16-e8-cc-20-90-00
```

The following is a sample display generated by the SHOW KNOWN CLEARINGHOUSES command:

```
Clearinghouse name ________________ mynode_ch
Clearinghouse state _______________ On
Clearinghouse filename ____________ sys$sysroot:[dns$server]mynode_ch.dns
RMS buffer count __________________ 3
Directories in this clearinghouse _ 55
UID _________________________aa-00-04-00-8e-11-a0-5a-1e-73-b4-6f-90-00
```

SHOW CLEARINGHOUSE: This command displays the following information for a clearinghouse:

- Clearinghouse name: The name that you specified during installation.
- Clearinghouse state: ON/OFF/INITIALIZING.
- Clearinghouse file name: The file specification for the clearinghouse file.
- RMS buffer count: The number of RMS buffers used for operation of the clearinghouse.
- Directories in this clearinghouse: The number of directories in this clearinghouse.
- UID: The unique identifier assigned to the clearinghouse when it was created.
- Next scheduled update: The absolute time when the next cleanup is scheduled to take place.
- Time since last zeroed: The amount of time that has elapsed since the counters were last zeroed.

NOTE

The following counters (except for the last three) are associated with DNS events, as described in Appendix C. The counters specify the number of times that the event occurred.

- Security violation: The number of times a security violation was detected.
- Tree broken: The number of times a child pointer could not be accessed from this DNS server. There seems to be a break in connections within the namespace.
- Root lost: The number of times a copy of a directory closer to the root than this one could not be found in this clearinghouse.
- Crucial replica: The number of attempts made to remove a crucial replica.
- Parent pointer: The number of times an unsuccessful attempt was made to update a child of this directory.
- Data corruption: The number of times an RMS error was detected when accessing the database.
- Possible cycle: The number of times a possible cycle was detected in following a chain of groups.
- Skulk failed: The number of times an update failed.
- Propagate failed: The number of times an update failed to propagate.
- Lookups satisfied: The number of times this clearinghouse satisfied a lookup request.
- Modifies satisfied: The number of times this clearinghouse satisfied a modify request.
- Requests assisted: The number of times this clearinghouse passed a request to another clearinghouse.

The following is a display returned by the SHOW CLEARINGHOUSE command:

```
DNS> SHOW CLEARINGHOUSE MYNODE_CH [RET]

Clearinghouse name ________________ mynode_ch
Clearinghouse state _______________ Off
Clearinghouse filename ____________ sys$sysroot:[dns$server]mynode_ch.dns
RMS buffer count __________________ 3
Directories in this clearinghouse _ 32
UID _______________________________ aa-00-04-00-8e-11-60-5a-0d-75-a9-6f-90·
Next scheduled update _____________ 2-OCT 1987 00:30:37.53
Time since last zeroed ____________ 0 05:14:16.88
Security violation ________________ 0
Tree broken _______________________ 0
Root lost _________________________ 0
Crucial replica ___________________ 0
Parent pointer ____________________ 0
Data corruption ___________________ 0
Possible cycle ____________________ 0
Skulk failed ______________________ 0
Propagate failed __________________ 0
Lookups satisfied _________________ 4
Modifies satisfied ________________ 0
Requests assisted _________________ 0
```

3.6.6 Setting the DNS Timezone

The DNS$CONTROL SET TIMEZONE command allows you to ensure that timestamp information generated by the DNS software on this node reflects Greenwich Mean Time (GMT). The value you enter with the SET TIMEZONE command is added to or subtracted from local time to reflect GMT.

As described in the *VAX Distributed Name Service Installation Guide* and in Chapter 4 of this manual, the value specified with this command must be in the range of -12 to +13 for hours, and 00-59 for minutes.

All system times and timezones for nodes using DNS must be within four minutes of each other. Otherwise, BADCLOCK errors occur and updates fail. See Appendix A for information on error messages.

Remember to change the timezone value for Daylight Savings Time and Standard Time, as described in the following section.

3.6.6.1 Resetting the DNS Timezone Value

To change the system time on DNS server nodes for Daylight Savings Time and Standard Time, do the following:

1. Stop each clearinghouse on the server by issuing the STOP CLEARINGHOUSE command.

   Do not stop the DNS server.
2. After stopping all the clearinghouses on the server, issue the SET TIMEZONE command to set the new timezone. If time goes one hour backward, the timezone is one hour further from GMT.

   For example, if you are on Eastern Daylight Savings Time, your summer timezone is -4:00 and your regular (or EST) timezone is -5:00.
3. Use the VMS SET TIME command to change the system time.
4. Restart each clearinghouse on the server by issuing the START CLEARINGHOUSE command.

3.6.7 Using Backup Copy of Database File

If you need to use the backup copy of the database file, issue the DNS$CONTROL REBUILD DIRECTORY command for each directory in the clearinghouse. This will ensure that updates made to any directories since the backup copy was made are in all copies of the directory. Before you issue the REBUILD DIRECTORY command, issue the SHOW CHARACTERISTICS CLEARINGHOUSE command to get a list of directories in that clearinghouse.

3.6.8 Maintaining and Tuning

This section tells how to improve DNS performance by optimizing RMS buffer counts of clearinghouses. It also tells you how to analyze information in the event log to maintain DNS.

3.6.8.1 Tuning

Optimizing the number of RMS buffers for a clearinghouse can improve DNS performance. You may want to perform the following procedures if you detect a decrease in DNS performance.

Invoke the RMS analyze program by entering the following at the DCL prompt:

ANALYZE/RMS/STATISTICS/OUTPUT=*filename*.DAT *dev*:[*dir*]*clearinghouse_filename.typ*

The resulting displays lists statistics for two area descriptors and for two keys. You can use these statistics to optimize the size of your RMS buffers.

The significant statistics are:

- Bucket size of your area descriptors
- Number of index levels
- Number of index blocks
- Number of data blocks

As a result of this analysis, perform one or both of the following procedures:

- Issue the DNS$CONTROL SET CLEARINGHOUSE command to modify the RMS buffer count. You must restart the clearinghouse for this to take effect.
- Issue the DNS$CONTROL REBUILD CLEARINGHOUSE command to reduce the number of index levels.

The following sections tell how to set your RMS buffers to cache all data and indexes.

To Cache All Data

Caching all data in the RMS buffers gives you better DNS performance but requires more virtual pages and larger enqueue quota.

To cache all data, do the following calculation. Then, issue the DNS$CONTROL SET CLEARINGHOUSE command to set the RMS buffer count to the resulting value.

RMS Buffer Count = ((Count of Index Blocks for Key #0 + Count of Data Blocks for Key #0)/(Bucket Size of Area Descriptor #0)) + ((Count of Index Blocks for Key #1 + Count of Data Blocks for Key #1)/(Bucket Size for Area Descriptor#1))

To Cache All Indexes

Caching all indexes in the buffer provides good DNS performance and requires fewer resources than caching all data.

To cache all indexes, do the following calculation. Then, issue the SET CLEARINGHOUSE command to set your RMS buffer count to the resulting value.

RMS Buffer Count = (Count of Index Blocks for Key #0/Bucket Size for Area Descriptor #0) + (Count of Index Blocks for Key #1/Bucket Size for Area Descriptor #1)

3.6.8.2 Using the Event Log

DNS uses the DECnet event logging facility to write DNS events to an output device. (Appendix C lists and describes all DNS events.) You can log DNS events to a file, console, or monitor. Information regarding the sequence and frequency of events can help determine whether you need to take maintenance action. Therefore, it is useful to keep the event logger on.

Issue the following Network Control Program (NCP) command to specify the component to which DNS events are logged:

NCP SET LOGGING *component* STATE ON

where the value for *component* is CONSOLE, MONITOR, or FILE. If you want to turn off the component, set the state to OFF.

For more information on NCP commands for event logging, see the *VAX/VMS Network Control Program Reference Manual.*

The DNS start-up command file, SYS$MANAGER:DNS$STARTUP.COM, enables logging for all events whose code begins with 352, and for events 353.1, 353.2, 353.3, and 353.4.

3.7 Distributing the Namespace

You distribute the namespace by copying directories from one clearinghouse to another. As described in Chapter 1, doing so increases the availability of DNS names and improves the performance of name lookup operations. This section describes how to distribute the namespace.

3.7.1 Installing Additional Servers

Each time you install a DNS server, a new clearinghouse is automatically created by the installation procedure.

Issue the DNS$CONTROL COPY DIRECTORY command to copy one or more directories to one or more clearinghouses in the namespace. Doing so allows you to replicate all or part of your namespace in any clearinghouse in your network.

3.7.2 Naming Clearinghouses

A clearinghouse is a file on a DNS server node in which DNS directories are stored. A **clearinghouse object** is an object in the namespace that contains name and address information about the clearinghouse. The DNS installation procedure creates both the clearinghouse file and the clearinghouse object.

The name you give to a clearinghouse during installation determines the location of the clearinghouse object in the namespace.

3.7.2.1 Specifying the Location of a Clearinghouse Object

During installation of your first DNS server, the clearinghouse object is stored in the root directory. When you install additional DNS servers that will participate in an existing, hierarchical namespace, you can locate the clearinghouse object in any directory that is allowed to contain it, as described in the next section. To do so, you specify a directory name in the format *directory-name.ch-name* in response to the installation prompt. The clearinghouse object is then stored in that directory.

3.7.2.2 Allowing Directories to Store Clearinghouse Objects

Directories that store clearinghouse objects use extra overhead during updates. Therefore, you may want to restrict the number of directories that can store clearinghouse objects. Doing so enables you to control and restrict the creation of clearinghouses and the growth of the namespace hierarchy.

You must allow a directory to store a clearinghouse object by issuing the DNS$CONTROL SET DIRECTORY command in the following format:

SET DIRECTORY *dir-name* ALLOW_CLEARINGHOUSE

If you do not want to allow a directory to store a clearinghouse object, enter the same in the following format:

SET DIRECTORY *dir-name* DISALLOW_CLEARINGHOUSE

The default is DISALLOW_CLEARINGHOUSE.

If you want to prohibit a directory from storing clearinghouse objects (which may be the case when you restructure the namespace), you must make sure that:

1. The directory does not contain any clearinghouse objects. To do so, issue the SHOW DIRECTORY command in the following format:

   SHOW DIRECTORY *dir-name* OBJECT * CLASS DNS$CLEARINGHOUSE

   This command returns the names of all clearinghouse objects stored in the directory. You cannot set the directory to DISALLOW_CLEARINGHOUSE until you delete all clearinghouse objects.

2. None of the children of that directory have been allowed to store clearinghouse objects. Issue the SHOW DIRECTORY command in the following format to display a list of the children in the directory:

   SHOW DIRECTORY *dir-name* CHILDREN *

   For each child directory displayed, issue the SHOW CHARACTERISTICS DIRECTORY COMMAND in the following format:

   SHOW CHARACTERISTICS DIRECTORY *dir-name*

   The resulting display tells you whether the directory is allowed to store clearinghouse objects. Then, for each allowed child, issue the SET DIRECTORY command in the following format:

   SET DIRECTORY *dir-name* DISALLOW

3.7.3 Creating Master and Read-Only Directories

There are two types of DNS directories:

1. Master directories
2. Read-only directories

A master directory is one that you create with the DNS$CONTROL CREATE DIRECTORY command. All changes to a directory must be made to the master directory.

A read-only directory is a copy of a master directory. You create a read-only directory with the DNS$CONTROL COPY DIRECTORY command. Applications cannot update read-only directories; they can only read them. Read-only directories are updated by the DNS update function, as described later in this chapter.

If you want to switch the location of a master directory with one of its read-only copies, use the DNS$CONTROL REBUILD DIRECTORY command, as described in Section 3.8.4.

3.7.4 Distributing Read-Only Directories to Multiple Clearinghouses

Issue the DNS$CONTROL COPY DIRECTORY command to distribute a read-only copy of a master directory to another clearinghouse. This is how you distribute your namespace.

In a single-directory namespace, the DNS installation procedure copies the root directory to another clearinghouse when installing multiple DNS servers. In a hierarchical namespace that has more than one DNS server, you can copy individual directories from one clearinghouse to another by issuing the COPY DIRECTORY command in the following format:

COPY DIRECTORY *dir-name* CLEARINGHOUSE *ch-name*

3.7.5 Updating Directories

The DNS update function ensures that any change (addition, deletion, or modification) made to a master directory is propagated to all read-only copies of that directory.

DNS automatically updates directories. It also allows you to force an update by issuing the DNS$CONTROL UPDATE DIRECTORY command. DNS tries to update of all copies of a directory immediately after a change is made to the master. In addition, DNS attempts to update all directories at regular intervals.

The frequency of DNS updates is controlled by the DNS$Convergence attribute for each directory. You set the convergence value by issuing the DNS$CONTROL SET DIRECTORY CONVERGENCE command. If you set the convergence value to LOW, DNS waits until the next scheduled update function to update all copies of the directory. If you set the convergence value to HIGH, DNS immediately tries to update directory copies.

The default value for the DNS$Convergence attribute is HIGH.

3.7.6 Clearinghouse Update

During DNS operation, clearinghouses become cluttered with deleted entries and attributes. These are automatically cleaned up during the scheduled update function.

You can reschedule the update function by issuing the DNS$CONTROL SET CLEARINGHOUSE TIMER command. You specify the hours and minutes from the current time that you want the update function to execute for a clearinghouse. You can also specify that you want the update to be attempted upon execution of the command.

To set a time interval for a clearinghouse update, issue the command in the following format:

SET CLEARINGHOUSE *ch-name* TIMER *hh:mm*

To execute the clearinghouse update immediately, issue the command in the following format:

SET CLEARINGHOUSE *ch-name* TIMER NOW

3.8 Restructuring the Namespace

A major restructuring of the namespace takes place when you change from a single-directory to a hierarchical format. You also restructure the namespace to keep it current with the organization it represents. For example, if departments within your organization are reorganized, the namespace structure should reflect those changes.

Another reason for restructuring the namespace is as a temporary measure to keep DNS available while nodes or lines are undergoing maintenance.

Restructuring the namespace may require you to perform the following tasks:

- Move or delete clearinghouses
- Link new names with old ones in the namespace
- Rebuild, remove, or delete directories

The following sections describe how to perform these tasks.

3.8.1 Moving a Clearinghouse

The DNS$CONTROL REMOVE CLEARINGHOUSE command removes a clearinghouse from the list of clearinghouses known to a particular DNS server so you can move the clearinghouse. Use this command in combination with the DNS$CONTROL ADD CLEARINGHOUSE command when you want to move the clearinghouse to a new disk or a new DNS server.

You may also want to move a clearinghouse to another node if its server node is being removed from the network permanently or taken down for a long maintenance period.

To move a clearinghouse, do the following:

1. Issue the following DNS$CONTROL commands in the following format:

   STOP CLEARINGHOUSE *ch-to-move*
   REMOVE CLEARINGHOUSE *ch-to-move*

2. Copy the clearinghouse file to its new location, that is, another disk or node. If the clearinghouse is being restored from a backup set to a new location, use the VMS BACKUP command to move the correct file to the new disk.

   The following is the command format for copying a clearinghouse file to a new directory on a new device:

   SYS$SYSROOT:[DNS$SERVER]*ch-to-move*.DNS
   new_dev:[*new_dir*]*.*

   You do not have to move the .GBL file for the clearinghouse. The ADD CLEARINGHOUSE command recreates it at the new location.

3. At the new location, issue the following command in the following format:

   ADD CLEARINGHOUSE *ch-to-move* FILE *new_dev*:[*new_dir*]*ch-to-move*.DNS

4. At the old location, you can now delete the .DNS and .GBL files for the clearinghouse.

3.8.2 Deleting a Clearinghouse

The DNS$CONTROL DELETE CLEARINGHOUSE command deletes a clearinghouse from the namespace. To use this command, the clearinghouse must not contain any master directories. To delete a clearinghouse, do the following:

1. Enter the following DNS$CONTROL command to display the names of all the directories in the clearinghouse that you want to delete:

   SHOW CHARACTERISTICS CLEARINGHOUSE *ch-name*

2. Enter the following command for each directory in the clearinghouse:

   SHOW CHARACTERISTICS DIRECTORY *dir-name*

   The resulting display tells you which clearinghouse contains the master copy of the directory.

3. Delete any master directories that are found. Before you delete a directory, you must empty it, as described in Section 3.8.6.

4. Enter the following command to delete the clearinghouse:

DELETE CLEARINGHOUSE *ch-name*

3.8.3 Creating and Deleting Links

A **link** is a pointer to the name of an existing entry in the namespace. Links can be useful when the name of a frequently used object is changed or when a namespace is restructured. Links are convenient for users who are accustomed to referring to an entry by its old name.

To change the name of an entry from A.B.C to X.Y.Z, delete entry A.B.C and create entry X.Y.Z. Then create the link from A.B.C to X.Y.Z by issuing the following command:

```
DNS> CREATE LINK A.B.C DESTINATION X.Y.Z [RET]
```

Subsequent references to entry A.B.C will be directed to entry X.Y.Z.

NOTE

> When you create a link, its unique identifier (UID) is displayed at your terminal.

You can optionally specify an expiration date and an extension time for the link; otherwise, the link is permanent.

When the expiration date passes, DNS checks whether the link destination exists. If it does not exist, DNS deletes the link. If it does exist, DNS adds the extension time value to the expiration value. If the extension value is equal to zero, DNS deletes the link.

To delete a link, you need to specify its name in the DNS$CONTROL DELETE LINK command.

The DNS$CONTROL SHOW KNOWN ATTRIBUTES and SHOW ATTRIBUTES commands enable you to display all known attributes or the value of a specific attribute for a link, respectively. Link attributes are defined in Appendix B.

3.8.4 Relocating a Master Directory to a New Clearinghouse

Use the DNS$CONTROL REBUILD DIRECTORY command to relocate a master directory from one clearinghouse to another in order to restructure the namespace.

To Relocate a Master Directory

Step 1:
Enter the SHOW CHARACTERISTICS DIRECTORY command in the following format to display a list of clearinghouses that contain copies of the directory:

SHOW CHARACTERISTICS DIRECTORY *dir-name*

Step 2:
Enter the REBUILD DIRECTORY command in the following format at the node associated with the directory copy that you want to change to a master:

REBUILD DIRECTORY *dir-name* MASTER *ch-name1* READ *ch-name2, ... ch-namen*

In this command, *dir-name* is the name of the directory whose master copy will be relocated; *ch-name1* is the name of the clearinghouse that will contain the new master copy. The parameters *ch-name2* to *ch-namen* are the remaining clearinghouses that were displayed in step 1.

The following is an example of the use of the REBUILD DIRECTORY command. Note that the SHOW CHARACTERISTICS DIRECTORY command is issued first to display a list of the clearinghouses that contain copies of the directory ENGINEERING.DEVELOPMENT.APPLICATIONS. In this example, the command is entered at node MYNODE.

```
DNS> SHOW CHARACTERISTICS DIRECTORY ENGINEERING.DEVELOPMENT.APPLICATIONS RET

    Clearinghouse______________________ALLOWED
    Convergence________________________HIGH
    Last successful update_____________11-SEP-1987 18:11:36.12

    Directory Copies:

       Address________________________YOURS (12.33)
       Replica Type___________________Master
       Clearinghouse__________________BANANA_NS:.YOURNODE_CH

       Address________________________MYNODE (5.32)
       Replica Type___________________Read-only
       Clearinghouse__________________BANANA_NS:.MYNODE_CH
```

```
          Address________________________NODEZ (4.28)
          Replica Type___________________Read-only
          Clearinghouse__________________BANANA_NS:.HISNODE_CH

DNS> REBUILD DIRECTORY ENGINEERING.DEVELOPMENT.APPLICATIONS MASTER MYNODE_CH
     READ YOURNODE_CH,HISNODE_CH RET
```

This REBUILD DIRECTORY command relocates the master copy of directory ENGINEERING.DEVELOPMENT.APPLICATIONS from clearinghouse YOURNODE_CH to clearinghouse MYNODE_CH and leaves HISNODE_CH as a read-only copy.

3.8.5 Removing Directory Copies from Permanently Unavailable Clearinghouses

When a clearinghouse becomes permanently unavailable, you may want to remove read-only directory copies from that clearinghouse. This is because you cannot guarantee that DNS will complete the update function for the master directories that have read-only directory copies in the permanently unavailable clearinghouse. DNS may begin the update function for a master directory, but when it tries to update the read-only copy in the unavailable clearinghouse, it will stop the update. Therefore, you cannot be sure that all read-only copies of the master directory are successfully updated.

To ensure that updates complete successfully in this case, issue the DNS$CONTROL REBUILD DIRECTORY command to remove the read-only directory copy in the unavailable clearinghouse from the set of read-only copies of the master directory. Do not use this command when a clearinghouse is unavailable for only a few hours. (To remove a directory from a clearinghouse under normal conditions, use the DNS$CONTROL REMOVE DIRECTORY command.)

In the example in Section 3.8.4, if the disk containing the clearinghouse HISNODE_CH was accidentally erased and if backups were not available, you would issue the following command (at node MYNODE) to repair the directory set:

```
DNS> REBUILD DIRECTORY ENGINEERING.DEVELOPMENT.APPLICATIONS MASTER
     MYNODE_CH READ YOURNODE_CH RET
```

Using the REBUILD DIRECTORY command to exclude a directory copy from a set of read-only directories does not delete the directory from the unavailable clearinghouse. The unavailable clearinghouse, however, should never reappear in the namespace. If it does reappear, you may get unpredictable responses to read requests for the excluded directory.

3.8.6 Deleting a Directory

You must meet two conditions before you can delete a directory from the namespace:

1. All read-only copies of the directory must be removed.
2. The directory must be empty. It must contain no child pointers, objects, groups, or links.

Emptying a Directory

This section tells how to empty a directory of child pointers, objects, groups, and links.

Enter the following commands in the formats shown and in the sequence specified.

Step 1:
Enter the following command, specifying the name of the directory that you want to delete:

```
SHOW CHARACTERISTICS DIRECTORY dir-name
```

This displays such information as the type (master or read-only) of each copy of the directory.

Step 2:
Enter the following command for each read-only copy of the directory you want to delete:

```
REMOVE DIRECTORY dir-name CLEARINGHOUSE ch-name
```

Step 3:
Enter the following commands for the directory that you want to delete:

```
SHOW DIRECTORY dir-name OBJECTS *
SHOW DIRECTORY dir-name LINKS *
SHOW DIRECTORY dir-name GROUPS *
```

Step 4:
Enter the following commands for each object, link, and group displayed in step 3:

```
DELETE OBJECT obj-name
DELETE LINK link-name
DELETE GROUP grp-name
```

Step 5:
Enter the following command for the directory that you want to delete:

```
DELETE DIRECTORY dir-name
```

3.8.7 Removing a Directory

If a read-only copy of a directory is no longer needed at a clearinghouse, you can remove the copy with the DNS$CONTROL REMOVE DIRECTORY command. You must stop the clearinghouse before you can remove the directory. To do so, issue the DNS$CONTROL STOP CLEARINGHOUSE command.

Chapter 4
DNS$CONTROL Commands

This chapter defines the formats and functions of the DNS$CONTROL management commands. To use these commands, you must first invoke DNS$CONTROL as described in Chapter 3.

The commands are described in alphabetical order, as follows:

- ADD

  ACCESS
  CLEARINGHOUSE
  MEMBER
- COPY DIRECTORY
- CREATE

  DIRECTORY
  GROUP
  LINK
- DELETE

  CLEARINGHOUSE
  DIRECTORY
  GROUP
  LINK
  OBJECT
- EXIT
- HELP
- REBUILD

  CLEARINGHOUSE
  DIRECTORY

- REMOVE

  ACCESS
  CLEARINGHOUSE
  DIRECTORY
  MEMBER

- SET

  CLEARINGHOUSE
  DIRECTORY
  TIMEZONE

- SHOW

  ACCESS
  ACTIVE CLEARINGHOUSES
  CHARACTERISTICS
  CHILD
  CLEARINGHOUSE
  DIRECTORY
  GROUP
  KNOWN CLEARINGHOUSES
  LINK
  NAMESERVER
  OBJECT
  VERSION

- START CLEARINGHOUSE
- STOP CLEARINGHOUSE
- UPDATE DIRECTORY

Access rights and system manager privileges are specified for each command. You must have the following privileges when system manager privileges are specified:

- NETMBX
- TMPMBX
- ACNT
- PRMGBL
- SHMEM
- ALTPRI
- OPER
- SYSLCK
- SYSGBL
- LOG_IO
- CMKRNL
- SYSNAM
- DETACH
- SYSPRV
- PFNMAP

ADD ACCESS

Adds an access control entry to a clearinghouse, directory, object, group, or link entry. Flags control how rights are granted to the contents of directories and whether the access is for a group or an individual. Access rights are contained in the DNS$ACS attribute.

Format

```
                        ( CLEARINGHOUSE ch-name )
                        | DIRECTORY dir-name    |   [ /FLAGS = (flags)   ]
ADD ACCESS   ace-name  <  GROUP grp-name         >  [ /RIGHTS = (rights) ]
                        | LINK link-name        |
                        ( OBJECT obj-name       )
```

Parameters

ace-name
Name of the individual or group being granted access. This name takes the form *node-name::user-name* or *grp-name.*

ch-name
Name of the clearinghouse to which you are adding access.

dir-name
Name of the directory to which you are adding access.

grp-name
Name of the group to which you are adding access.

link-name
Name of the link to which you are adding access.

obj-name
Name of the object to which you are adding access.

flags

Any or all of the following:

NOPROPAGATE: Use only for controlling access to a directory. Specifies that the access rights granted are not to be inserted into the access control set of new child directories. If you do not enter a value, the specified rights will propagate to child directories created in this directory. This flag does not affect access rights to existing child directories.
DEFAULT: Use only for adding access to the contents of a directory. Any access entry with this flag is passed on to new objects, groups, and links created in that directory. This flag does not affect access rights to existing objects, groups, and links. It also does not grant or deny access to the directory itself.
GROUP: Use to indicate that the *ace-name* specified in the command is an explicit group.

rights

Any or all of the following:

READ: Enables *ace-name* to examine an entry.
WRITE: Enables *ace-name* to change the content of an entry.
DELETE: Enables *ace-name* to delete an entry.
TEST: Enables *ace-name* to test a specified value against the actual value of an attribute.
CONTROL: Enables *ace-name* to modify the access control set.

You can also enter NONE to deny access to *ace-name*.

Examples

1.
```
DNS> ADD ACCESS MYNODE::SPEARS DIRECTORY ENGINEERING.DEVELOPMENT
/FLAGS=(DEFAULT) /RIGHTS=(READ,WRITE,TEST,CONTROL,DELETE)
```

This command adds an access control entry to the access control set of directory ENGINEERING.DEVELOPMENT. The DEFAULT flag indicates that this access control entry will be applied to newly created objects or links in the ENGINEERING.DEVELOPMENT directory. User MYNODE::SPEARS will have read, write, test, control, and delete access to newly created objects in this directory. This command does not give user MYNODE::SPEARS access to directory ENGINEERING.DEVELOPMENT.

ADD ACCESS

Required Access Rights

To use this command, you (expressed as *node-name*::*user-name*) need control access to the entry to which access is being added.

ADD CLEARINGHOUSE

Adds an existing clearinghouse to a DNS server. The REMOVE CLEARINGHOUSE operation must first have been performed at the old location of the clearinghouse. This combination of commands is generally used to move clearinghouses from one node to another, or from one disk to another on the same node.

Format

ADD CLEARINGHOUSE *ch-name* FILE *file-name*

Parameters

ch-name
Name of the clearinghouse to be added.

file-name
Name of the file that contains the clearinghouse database.

Examples

1.
```
DNS> ADD CLEARINGHOUSE SPEARS_CH FILE SYS$SPECIFIC:
[DNS$SERVER]:SPEARS_CH.DNS
```

This command adds the clearinghouse SPEARS_CH, contained in the file SYS$SPECIFIC:SPEARS_CH.DNS to the DNS server.

Required Access Rights

- To use this command, you (expressed as *node-name::user-name*) need read, write, delete, and test access to the directory that stores the clearinghouse object. In the example, this would be the root directory.
- Requires system manager privileges.

ADD MEMBER

Adds an individual member or a group to a group. Members are stored in the DNS$Members attribute of the group.

Format

ADD MEMBER *mem-name* GROUP *grp-name* [/FLAGS=(GROUP)]

Parameters

mem-name
Name of the member to be added.

grp-name
Name of the group to which the member is added.

Qualifier

/FLAGS=(GROUP)
Specify only if the member to be added is a group.

Examples

1.
```
DNS> ADD MEMBER BOSTON::D_SMITH GROUP HOME.OFFICE_PERSONNEL
```
This command adds member BOSTON::D_SMITH to group HOME.OFFICE_PERSONNEL.

2.
```
DNS> ADD MEMBER HOME.OFFICE_PERSONNEL GROUP ADMINISTRATION.HQ
/FLAG=(GROUP)
```
This command adds member HOME.OFFICE_PERSONNEL (which is a group) to group ADMINISTRATION.HQ.

Required Access Rights

To use this command, you (expressed as *node-name::user-name*) need write access to the group. In Example 1, the group would be HOME.OFFICE_PERSONNEL; in Example 2, the group would be ADMINISTRATION.

COPY DIRECTORY

Adds a copy of a directory to a clearinghouse.

Format

COPY DIRECTORY *dir-name* CLEARINGHOUSE *ch-name*

Parameters

dir-name
Name of the directory to be copied.

ch-name
Name of the clearinghouse to which the directory will be copied.

Examples

1.
```
DNS> COPY DIRECTORY SALES.CREDIT CLEARINGHOUSE HOME.OFFICE_CREDIT_CH
```

This command creates a copy of directory SALES.CREDIT in clearinghouse HOME.OFFICE_CREDIT_CH.

Required Access Rights

- To use this command, you (expressed as *node-name*::*user-name*) need:
  - Control access to *dir-name*. In the example, this would be SALES.CREDIT.
  - Write access to *ch-name*. In the example, this would be HOME.OFFICE_CREDIT_CH.

- The node that contains CLEARINGHOUSE *ch-name* (*node*::DNS$SERVER) needs:
  - Read, write, delete, and control access to *dir-name*. In the example, this would be SALES.CREDIT.
  - Write access to the parent directory. In the example, this would be SALES.

CREATE DIRECTORY

Creates a directory and places it in a clearinghouse that you specify.

Format

CREATE DIRECTORY *dir-name* [CLEARINGHOUSE *ch-name*]

Parameters

dir-name
Name of the directory you are creating.

ch-name
Name of the clearinghouse in which you want to place the directory. If you do not specify this name, DNS places the directory in the clearinghouse that stores the master copy of the parent directory.

Examples

1.
```
DNS> CREATE DIRECTORY BOSTON.SALES.87 CLEARINGHOUSE BOSTON.BRANCH_CH
```

This command creates a directory named BOSTON.SALES.87 and stores it in clearinghouse BOSTON.BRANCH_CH.

Required Access Rights

- To use this command, you (expressed as *node-name*::*user-name*) need:
  - Write access to *ch-name*. In the example, this would be BOSTON.BRANCH.
  - Write access to the parent directory. In the example, this would be BOSTON.SALES.
- The server where *ch-name* is located (*node*::DNS$SERVER) needs write access to parent directory. In the example, this would be BOSTON.SALES.

CREATE GROUP

Creates a group to be used for access control. Use the ADD MEMBER command to add members to the group.

Format

CREATE GROUP *grp-name*

Parameters

grp-name
Name of the group to be created.

Examples

1.
   ```
   DNS> CREATE GROUP HOME.OFFICE_PERSONNEL
   ```

   This command creates a group named HOME.OFFICE_PERSONNEL.

Required Access Rights

To use this command, you (expressed as *node-name::user-name*) need write access to the directory that stores the group name. In the example, this would be HOME.

CREATE LINK

Creates a link to another namespace entry. Also, optionally specifies the date and time for link expiration and an extension to that date and time. The extension is used if, upon expiration, the link destination still exists. If you do not include expiration or extension values in the command, the link is permanent until you delete it.

Format

```
                                                 [ EXPIRATION exp-time ]
CREATE LINK  link-name DESTINATION dest-name     [ EXTENSION ext-time  ]
```

Parameters

link-name
Name of the link to be created.

dest-name
Name of the entry to which *link-name* points.

exp-time
Date and time of expiration in the following VMS date/time format: *dd-mmm-yyyy hh:mm:ss.*

ext-time
Time that you want to add to *exp-time* if *dest-name* still exists at *exp-time*. The value of *ext-time* is expressed in the following VMS delta-time format: *dddd hh:mm:ss.*

Examples

1.
```
DNS> CREATE LINK BOS.MEMDEST DESTINATION BOSTON.OFFICE.MEMOS
```

This command creates the link BOS.MEMDEST that points to destination BOSTON.OFFICE.MEMOS. Because neither the expiration time nor the extension time is specified, this link can be deleted only with the DELETE LINK command.

2.
```
DNS> CREATE LINK BOS.MEMDEST DESTINATION BOSTON.OFFICE.MEMOS EXPIRATION
24-DEC-1987 23:59:59
```

This command creates the link BOS.MEMDEST that points to destination BOSTON.OFFICE.MEMOS. It specifies the time at which the link will expire.

3.
```
DNS> CREATE LINK BOS.MEMDEST DESTINATION BOSTON.OFFICE.MEMOS EXPIRATION
24-DEC-1987 23:59:59 EXTENSION 0007 00:00:01
```

This command creates the link BOS.MEMDEST that points to destination BOSTON.OFFICE.MEMOS. It specifies the time at which the link will expire. At that time, if BOSTON.OFFICE.MEMOS still exists, the specified extension value is added to the current expiration date and time to form a new expiration time. The extension time is retained.

Required Access Rights

To use this command, you (expressed as *node-name::user-name*) need write access to the directory in which *link-name* will be stored. In the examples, this is BOS.

DELETE CLEARINGHOUSE

Deletes a clearinghouse from the namespace. The clearinghouse must be empty of all master directory names before you can execute this command. This command attempts to remove all read-only directories first.

Format

DELETE CLEARINGHOUSE *ch-name*

Parameters

ch-name
Name of the clearinghouse to be deleted.

Examples

1.
```
DNS> DELETE CLEARINGHOUSE ACCOUNTING_CH
```

This command deletes a clearinghouse named ACCOUNTING_CH.

Required Access Rights

To use this command, you (expressed as *node-name::user-name*) need delete access to the clearinghouse object and system manager privileges. (The clearinghouse object in the example is ACCOUNTING_CH.)

If there are any read-only copies of a directory in the clearinghouse, you (expressed as *node-name::user-name*) need control access to the directory and write access to the clearinghouse.

DELETE DIRECTORY

Deletes a directory. The directory must be empty before you can delete it. In addition, there must be no copies of it in any other clearinghouse.

Format

DELETE DIRECTORY *dir-name*

Parameters

dir-name
Name of the directory to be deleted.

Examples

1.
```
DNS> DELETE DIRECTORY FINANCE.ACCOUNTS_RECEIVABLE
```

   This command deletes a directory named FINANCE.ACCOUNTS_RECEIVABLE.

Required Access Rights

- To use this command, you (expressed as *node-name*::*user-name*) need:
  - Write access to the clearinghouse that stores the directory.
  - Delete access to *dir-name*.
- The server on which the clearinghouse is located (*node*::DNS$SERVER) requires delete access to the parent directory. In the example, the parent directory is FINANCE.

DELETE GROUP

Deletes a group. It does not remove the group from access control entries that refer to it.

Format

DELETE GROUP *grp-name*

Parameters

grp-name
Name of the group to be deleted.

Examples

1.
```
DNS> DELETE GROUP SALES:PERSONNEL
```

   This command deletes the group named SALES.PERSONNEL.

Required Access Rights

To use this command, you (expressed as *node-name::user-name*) need delete access to *grp-name*.

DELETE LINK

Deletes a link from the namespace.

Format

DELETE LINK *link-name*

Parameters

link-name
Name of the link to be deleted.

Examples

1.
```
DNS> DELETE LINK BOS.MEM
```

   This command deletes link BOS.MEM from the namespace.

Required Access Rights

To use this command, you (expressed as *node-name::user-name*) need delete access to *link-name*.

DELETE OBJECT

Deletes an object from the namespace.

Format

DELETE OBJECT *obj-name*

Parameters

obj-name
Name of the object to be deleted.

Examples

1.
```
DNS> DELETE OBJECT ENGINEERING.DEVELOPMENT.OLD_STUFF
```

This command deletes an object called OLD_STUFF that is located in directory ENGINEERING.DEVELOPMENT.

Required Access Rights

To use this command, you (expressed as *node-name::user-name*) need delete access to *obj-name*. In the example, this would be ENGINEERING.DEVELOPMENT.OLD_STUFF.

EXIT

Exits DNS$CONTROL and returns the VMS dollar sign ($) prompt. You can also exit DNS$CONTROL by pressing CTRL/Z.

Format

EXIT

HELP

Displays information about DNS$CONTROL commands on your screen.

Format

```
HELP   ┌ command                    ┐
       │ command-level [wildcard]   │
       └ command-keyword            ┘
```

Parameters

command
Command for which you want information.

command-level
Keyword, or topic, or subtopic.

wildcard
An optional entry represented by an asterisk (*). If you specify an asterisk after the keyword, topic, or subtopic, you will get all the information available for those categories.

command-keyword
Any keyword applicable to the command.

Examples

1. `DNS> HELP ADD`

   This displays information on the function of the ADD command as applied to ACCESS, MEMBERS, and GROUPS. It also points to subtopics.

2. `DNS> HELP ADD ACCESS`

   This displays overview information on the ADD ACCESS command. It also indicates that there is another level of information available on flags and rights.

3. `DNS> HELP ADD ACCESS *`

   This displays information on the flags, rights, and formats for the ADD ACCESS command. It also displays examples.

Required Access Rights

None.

REBUILD CLEARINGHOUSE

Performs an RMS convert operation on a clearinghouse file. It reclaims unused space in the clearinghouse that usually results from deletions and modifications. A new version of the file is created; the old version can be deleted. The clearinghouse is stopped before this operation and restarted upon completion.

CAUTION

> Disk space equal in size to the clearinghouse being rebuilt must be available in order for this operation to complete successfully.

Format

REBUILD CLEARINGHOUSE *ch-name*

Parameters

ch-name
Name of the clearinghouse that you want to rebuild.

Examples

1. `DNS> REBUILD CLEARINGHOUSE ACCOUNTING.PERSONNEL`

   This command performs an RMS convert operation on the file for clearinghouse ACCOUNTING.PERSONNEL.

Required Access Rights

No specific access rights are needed. Requires system manager privileges.

REBUILD DIRECTORY

Performs either or both of the following functions:

- Excludes one or more read-only copies of a master directory from the set of directory copies for the master directory.
- Redesignates a directory type from read-only to master.

When changing a directory from a read-only copy to a master, the REBUILD DIRECTORY command must be executed on the DNS server that will contain the master copy of the directory.

Format

REBUILD DIRECTORY *dir-name* MASTER *ch-master* [read-only *ch-1,..ch-n*]

Parameters

dir-name
Name of the directory you are rebuilding.

ch-master
Name of the clearinghouse in which the name of the master directory will be stored.

ch-1,..ch-n
Names of the clearinghouses in which read-only copies of *dir-name* will be stored. Clearinghouse names must be separated by commas.

Examples

1.
```
DNS> REBUILD DIRECTORY SALES.1987 MASTER HOME.OFFICE_SALES
read-only SALES.BOSTON,SALES.TULSA,SALES.ZURICH
```

This command rebuilds the set of copies of directory SALES.1987. It also designates the copy at clearinghouse HOME.OFFICE_SALES as the master and the copies in clearinghouses SALES.BOSTON, SALES.TULSA, and SALES.ZURICH as read-only copies.

NOTE

A copy of directory SALES.1987 must exist in each of the clearinghouses.

Required Access Rights

- To use this command, you (expressed as *node-name::user-name*) need read and control access to the directory. In the example, this is SALES.1987
- To change the type of directory, you must have write access to the clearinghouse that stores the directory whose type is being changed.

REMOVE ACCESS

Removes an access control entry from the access control set of a clearinghouse, directory, object, group, or link.

Format

```
                        ( CLEARINGHOUSE ch-name )
                        | DIRECTORY dir-name    |
REMOVE ACCESS  ace-name < GROUP grp-name        > [/FLAGS=(DEFAULT)]
                        | LINK link-name        |
                        ( OBJECT obj-name       )
```

Parameters

ace-name
Name of the access control entry in the form *node-name::user-name,* or a group name.

ch-name
Name of the clearinghouse from which you are removing access.

dir-name
Name of the directory from which you are removing access.

grp-name
Name of the group from which you are removing access.

link-name
Name of the link from which you are removing access.

obj-name
Name of the object from which you are removing access.

Examples

1.
```
DNS> REMOVE ACCESS MYNODE::SPEARS DIRECTORY ENGINEERING.DEVELOPMENT
```

This command removes access rights for MYNODE::SPEARS from directory ENGINEERING.DEVELOPMENT.

2.
```
DNS> REMOVE ACCESS HOME.OFFICE_PERSONNEL DIRECTORY
ENGINEERING.DEVELOPMENT /FLAGS=(DEFAULT)
```

This command removes access rights for group HOME.OFFICE_PERSONNEL from directory ENGINEERING.DEVELOPMENT.

Required Access Rights

To use this command, you (expressed as *node-name*::*user-name*) need control access to the entry from which you are removing access.

REMOVE CLEARINGHOUSE

Removes a clearinghouse from a DNS server so that the clearinghouse file can be moved. The clearinghouse can be moved to a new disk on the same node or to another DNS server. The clearinghouse must be off when you issue this command.

Format

REMOVE CLEARINGHOUSE *ch-name*

Parameters

ch-name
Name of clearinghouse to be removed.

Examples

1.
```
DNS> REMOVE CLEARINGHOUSE MYNODE_CH
```

   This command removes a clearinghouse named MYNODE_CH from the list of clearinghouses on the DNS server.

Required Access Rights

No specific access rights. Requires system manager privileges.

REMOVE DIRECTORY

Removes a read-only copy of a directory from a clearinghouse.

Format

REMOVE DIRECTORY *dir-name* CLEARINGHOUSE *ch-name*

Parameters

dir-name
Name of the directory.

ch-name
Name of the clearinghouse from which the directory copy is to be removed.

Examples

1.
```
DNS> REMOVE DIRECTORY ENGINEERING.DEVELOPMENT CLEARINGHOUSE OURNODE_CH
```

This command removes a read-only copy of directory ENGINEERING.DEVELOPMENT from a clearinghouse named OURNODE_CH.

Required Access Rights

To use this command, you (expressed as *node-name::user-name*) need control access to the directory and write access to the clearinghouse.

REMOVE MEMBER

Removes a member from a group. The member can be an individual or a group.

Format

REMOVE MEMBER *mem-name* GROUP *grp-name* [/FLAGS=(GROUP)]

Parameters

mem-name
Name of the member or group to be removed from the group. This is expressed as *node-name*::*user-name* for an individual, and as a group name if the member is a group.

grp-name
Name of the group from which the member is to be removed.

Command Qualifiers

/FLAGS=(GROUP)
Specify only if the member being removed is a group.

Examples

1.
```
DNS> REMOVE MEMBER BOSTON::D_SMITH GROUP HOME.OFFICE_PERSONNEL
```
This command removes member BOSTON::D_SMITH from group HOME.OFFICE_PERSONNEL

2.
```
DNS> REMOVE MEMBER HOME.OFFICE_PERSONNEL GROUP ADMINISTRATION.HQ
/FLAGS=(GROUP)
```
This command removes group member HOME.OFFICE_PERSONNEL from group ADMINISTRATION.HQ.

Required Access Rights

To use this command, you (expressed as *node-name*::*user-name*) need write access to group. In Example 1, this would be HOME_OFFICE.PERSONNEL. In Example 2, it would be ADMINISTRATION.HQ.

SET CLEARINGHOUSE

Performs two distinct functions, depending on the format you use.

- Modifies the RMS buffer count of a clearinghouse.

  This function doesn't take effect until the clearinghouse is stopped and restarted.
- Causes a cleanup function to execute at a clearinghouse after a specified length of time has elapsed or as soon as you enter the command.

Format

$$\textbf{SET CLEARINGHOUSE}\quad \textit{ch-name} \left\{ \begin{array}{l} \text{RMS\_BUFFER\_COUNT } \textit{num-bufs} \\ \text{TIMER} \left\{ \begin{array}{l} \textit{hh:mm} \\ \text{NOW} \end{array} \right\} \end{array} \right\}$$

Parameters

ch-name
Name of the clearinghouse on which to perform the operation.

num-bufs
Number of RMS buffers to use, expressed in number of RMS buckets. The range is from 3 to 127.

hh:mm
Hours and minutes from the current time at which you want the update procedure to execute. The range for *hh* is from 00 to 12; the range for *mm* is from 00 to 59.

Examples

1.
```
DNS> SET CLEARINGHOUSE MYNODE_CH RMS_BUFFER_COUNT 100
```

   This command sets the RMS buffer count for clearinghouse MYNODE_CH to 100 buckets.

2. `DNS> SET CLEARINGHOUSE MYNODE_CH TIMER 02:30`

   This command schedules a cleanup of clearinghouse MYNODE_CH two and a half hours from now.

3. `DNS> SET CLEARINGHOUSE MYNODE_CH TIMER NOW`

   This command causes a cleanup of clearinghouse MYNODE_CH now.

Required Access Rights

No specific access rights are needed. Requires system manager privileges.

SET DIRECTORY

Performs two distinct functions, depending on the format format you use.

- Establishes or modifies the value of the DNS$InCHName attribute to specify whether a directory is allowed to store clearinghouse objects.
- Establishes or modifies the value of the DNS$Convergence attribute, thereby specifying the level of persistence with which the DNS will try to update copies of the designated directory.

Format

```
                             { { ALLOW_CLEARINGHOUSE     } }
                             { { DISALLOW_CLEARINGHOUSE  } }
SET DIRECTORY     dir-name   {                             }
                             { CONVERGENCE { HIGH }        }
                             {             { LOW  }        }
```

Parameters

dir-name
Name of the directory on which to perform the operation.

Arguments

ALLOW_CLEARINGHOUSE
Specifies that *dir-name* is allowed to store clearinghouse objects.

DISALLOW_CLEARINGHOUSE
Specifies that *dir-name* is not allowed to store clearinghouse objects. This is the default value.

CONVERGENCE
Specifies that the command is to modify the value of the DNS$Convergence attribute.

HIGH

Specifies that the update function should try to update all copies of *dir-name* updated as changes are made to the master. The update will execute every 12 hours. HIGH is the default.

LOW

Specifies that the update function executes at least once every 24 hours.

Examples

1.
```
DNS> SET DIRECTORY ENGINEERING.RESEARCH ALLOW_CLEARINGHOUSE
```

   This command sets the DNS$InCHName attribute value so that directory ENGINEERING.RESEARCH can store clearinghouse objects.

2.
```
DNS> SET DIRECTORY ENGINEERING.DEVELOPMENT CONVERGENCE HIGH
```

   This command sets the DNS$Convergence attribute value so that the update function tries to update copies of directory ENGINEERING.DEVELOPMENT immediately (that is, as changes are made to the master).

Required Access Rights

- For ALLOW_CLEARINGHOUSE or DISALLOW_CLEARINGHOUSE: You, expressed as *node-name*::*user-name* need control access to the directory. In Example 1, this is ENGINEERING.RESEARCH.
- For CONVERGENCE: You, expressed as *node-name*::*user-name* need write access to directory. In Example 2, this is ENGINEERING.DEVELOPMENT.

SET TIMEZONE

Specifies the offset between your local time and Greenwich Mean Time (GMT). Use this command only when you need to change the local time (using the DCL SET TIME command) because of a change in Daylight Savings Time.

NOTE

> You must stop the clearinghouse, change the system time, set the timezone, and then restart the clearinghouse.

Format

SET TIMEZONE $\left\{\begin{matrix} + \\ - \end{matrix}\right\}$ *hh:mm*

Parameters

hh:mm

Hours and minutes that, when added to or subtracted from local time, will give you Greenwich Mean Time (GMT). The range of values for *hh* is from -12 to + 13; the range of values for *mm* is from 00 to 59. Specify a positive value if the node is east of the Greenwich meridian. Specify a negative value if the node is west of the Greenwich meridian.

Examples

1.
```
DNS> SET TIMEZONE --02:00
```

This command is appropriate for a node west of the Greenwich meridian, where local time is two hours behind GMT.

SET TIMEZONE

Required Access Rights

No specific access rights are needed. Requires system manager privileges.

NOTE

Make sure that when setting time backwards, you set both the system time and the timezone. If you do not, you will inhibit DNS activity until the combination of system and timezone time catch up with the last used UID.

SHOW ACCESS

Displays the access rights for a clearinghouse, directory, group, link, or object.

Format

```
             ⎧ CLEARINGHOUSE ch-name ⎫
             ⎪ DIRECTORY dir-name    ⎪
SHOW ACCESS  ⎨ GROUP grp-name        ⎬
             ⎪ LINK link-name        ⎪
             ⎩ OBJECT obj-name       ⎭
```

Parameters

ch-name
Name of the clearinghouse whose access rights you want to display.

dir-name
Name of the directory whose access rights you want to display.

grp-name
Name of the group whose access rights you want to display.

link-name
Name of the link whose access rights you want to display.

obj-name
Name of the object whose access rights you want to display.

Examples

1.
```
DNS> SHOW ACCESS DIRECTORY ENGINEERING.DEVELOPMENT
```

This command displays the access rights for directory ENGINEERING.DEVELOPMENT.

Required Access Rights

Requires read access to entry.

SHOW ACTIVE CLEARINGHOUSES

Displays information for each active clearinghouse. The information displayed includes:

- Clearinghouse name
- Clearinghouse state
- Clearinghouse file name
- RMS buffer count
- Number of directories in this clearinghouse
- Clearinghouse UID

Format

SHOW ACTIVE CLEARINGHOUSES

Required Access Rights

No access rights are required. To use this command, you must have NETMBX and TMPMBX privileges.

SHOW CHARACTERISTICS

Displays groups of attribute values for specific entries.

Format

```
                          ( CHILD child-name               )
                          | CLEARINGHOUSE ch-name          |
SHOW CHARACTERISTICS      { DIRECTORY dir-name             }
                          | GROUP grp-name                 |
                          | LINK link-name                 |
                          ( OBJECT obj-name                )
```

Parameters

child-name
Name of the child pointer for which you want to display attribute information. Displays the node name, address, and name of the clearinghouse that stores the directory copy, and the type of copy at that clearinghouse (DNS$Replicas attribute).

ch-name
Name of the clearinghouse for which you want to display attribute information. Displays all directory copies in the clearinghouse (DNS$CHDirectories attribute).

dir-name
Name of the directory for which you want to display attribute information. Displays the node name, address and clearinghouse name for each copy of the directory (DNS$Replicas attribute); the last successful update of the directory (DNS$AllUpTo attribute); the persistence (HIGH or LOW) with which the update function tries to update copies of the directory (DNS$Convergence attribute); and whether the directory can store clearinghouse objects (DNS$InCHName attribute).

grp-name
Name of the group for which you want to display attribute information. (A group is a type of object.) Displays a list of group members (DNS$Members attribute).

link-name
Name of the link for which you want to display attribute information. Displays the name of the entry the link points to (DNS$LinkTarget attribute) and the expiration and extension times (DNS$LinkTimeout attribute).

obj-name
Name of the object for which you want to display attribute information. Displays the class to which the object belongs (DNS$Class attribute), the version assigned to it when it was created (DNS$ClassVersion attribute) and an address, if one has been assigned (DNS$Address attribute).

Examples

1. `DNS> SHOW CHARACTERISTICS CLEARINGHOUSE MYNODE_CH`

   This command displays the names of all directories stored in clearinghouse MYNODE_CH.

Required Access Rights

Requires read access to the entry.

SHOW CHILD

Displays the names of all attributes associated with a child pointer or the value of a specific attribute of a child pointer. See Appendix B for a list of all attributes.

Format

SHOW CHILD *child-name* $\left\{\begin{array}{l}\text{KNOWN ATTRIBUTES}\\ \text{ATTRIBUTE } \textit{att-name}\end{array}\right\}$

Parameters

child-name
Name of the child pointer.

att-name
Name of the attribute whose value you want to display.

Examples

1.
```
DNS> SHOW CHILD A.B KNOWN ATTRIBUTES
```
This command displays the names of all the attributes associated with child pointer A.B.

2.
```
DNS>SHOW CHILD A.B ATTRIBUTE DNS$REPLICAS
```
This command displays the value of attribute DNS$Replicas for child pointer A.B. It will indicate the location of all the copies of directory A.B.

Required Access Rights

To use this command, you (expressed as *node-name::user-name*) need read access to the parent entry.

SHOW CLEARINGHOUSE

Displays counter information, names of attributes, or the value of a specified attribute for a clearinghouse. See Appendix B for attribute information.

Format

SHOW CLEARINGHOUSE *ch-name* { ATTRIBUTE *att-name* | KNOWN ATTRIBUTES }

Parameters

ch-name
Name of the clearinghouse for which you want attribute information.

att-name
Name of the attribute whose value you want to display. See Appendix B for a list of attributes.

Examples

1.
```
DNS> SHOW CLEARINGHOUSE ACCOUNTING.OFFICE
```

This command displays state information and counter statistics for clearinghouse ACCOUNTING.OFFICE.

2.
```
DNS> SHOW CLEARINGHOUSE ACCOUNTING.OFFICE KNOWN ATTRIBUTES
```

This command displays the names of the known attributes for clearinghouse ACCOUNTING.OFFICE.

3.
```
DNS> SHOW CLEARINGHOUSE ACCOUNTING.OFFICE ATTRIBUTE DNS$NSUID
```

This command displays the value of the DNS$NSUID attribute for clearinghouse ACCOUNTING.OFFICE.

Required Access Rights

For Example 1, no specific access rights are required. Need NETMBX and TMPMBX privileges.

For Examples 2 and 3, you (expressed as *node-name*::*user-name*) need read access to the clearinghouse.

SHOW DIRECTORY

Displays any of the following types of information for a directory.

- Value of one or all directory attributes
- Information on all links or on a specified link stored in the directory
- Information on all child pointers or on a specified child pointer stored in the directory
- Information on all objects or on a specified object stored in the directory

See Appendix B for attribute information.

Format

```
                               ⎧ ATTRIBUTE att-name                                                ⎫
                               ⎪ KNOWN ATTRIBUTES                                                  ⎪
                               ⎪ LINK ⎧ link-name      ⎫                                           ⎪
                               ⎪      ⎩ wildcard-name  ⎭                                           ⎪
SHOW DIRECTORY    dir-name     ⎨ CHILDREN ⎧ child-name     ⎫                                       ⎬
                               ⎪          ⎩ wildcard-name  ⎭                                       ⎪
                               ⎪        ⎧ obj-name                                       ⎫         ⎪
                               ⎪ OBJECT ⎨ wildcard-name ⎡CLASS ⎧ child-name     ⎫⎤       ⎬         ⎪
                               ⎩        ⎩               ⎣      ⎩ wildcard-name  ⎭⎦       ⎭         ⎭
```

Parameters

dir-name
Name of the directory for which you want to display information.

att-name
Name of the directory attribute for which you want to display values. See Appendix B for a list of all attributes.

link-name
Name of the link in the directory for which you want to display information.

child-name
Name of the child directory in the directory for which you want to display information.

obj-name
Name of the object in the directory for which you want to display information.

class-name
Class name of the object in the directory for which you want to display information. This entry is optional.

wildcard-name
Wildcard specification denoted by an asterisk (*) or a question mark (?). Displays all or selected information. If specified without the name of a link, child, or object, displays information on all links, children, or objects in the directory. The question mark replaces only one character in the name specification. The asterisk can represent a string of characters. For example, A?C returns all three-character names beginning with A and ending with C. A*Z represents all names beginning with A and ending with Z.

Examples

1. `DNS> SHOW DIRECTORY ACCOUNTING.OFFICE`

   This command displays the names of all attributes of directory ACCOUNTING.OFFICE.

2. `DNS> SHOW DIRECTORY ACCOUNTING.OFFICE ATTRIBUTE DNS$LASTSKULK`

   This command displays the timestamp of the last update procedure on directory ACCOUNTING.OFFICE.

3. `DNS> SHOW DIRECTORY ACCOUNTING.OFFICE OBJECT D*`

   This command displays information on all objects beginning with the character D that are stored in directory ACCOUNTING.OFFICE.

4. `DNS> SHOW DIRECTORY ACCOUNTING.OFFICE KNOWN ATTRIBUTES`

   This command displays information on all attributes of directory ACCOUNTING.OFFICE.

5.
```
DNS> SHOW DIRECTORY ACCOUNTING.OFFICE OBJECT *
```

   This command displays information on all objects in directory ACCOUNTING.OFFICE.

Required Access Rights

- To use this command, you (expressed as *node-name::user-name*) need read access to directory.
- For links and objects, this command displays only the entries to which the user has access.

SHOW GROUP

Displays any of the following:

1. Names of all members of a group
2. Names of all attributes of a group
3. Value of a specific attribute of a group
4. Information that tells you whether a specific name is a member of a group

See Appendix B for attribute information.

Format

```
                          ⎧ ATTRIBUTE att-name  ⎫
SHOW GROUP   grp-name     ⎨ KNOWN ATTRIBUTES    ⎬
                          ⎪ KNOWN MEMBERS       ⎪
                          ⎩ MEMBER mem-name     ⎭
```

Parameters

grp-name
Name of the group whose attributes or members you want to display.

att-name
Name of the attribute whose value you want to display.

mem-name
Name that you are testing to determine its membership in the specific group. This is an individual name, expressed as (*node-name*::*user-name*) or a group name.

Examples

1. `DNS> SHOW GROUP SALES.TEAM KNOWN MEMBERS`

   This command displays the known members of the group SALES.TEAM.

2. `DNS> SHOW GROUP SALES.TEAM MEMBER D_SMITH`

   This command displays information that tells you whether D_SMITH is a member of the group SALES.TEAM.

3. `DNS> SHOW GROUP SALES.TEAM ATTRIBUTE DNS$ACS`

   This command displays the values of the DNS$ACS attribute for the group SALES.TEAM.

4. `DNS> SHOW GROUP SALES.TEAM KNOWN ATTRIBUTES`

   This command displays the names of all the attributes for group SALES.TEAM.

Required Access Rights

- To use this command, you (expressed as *node-name*::*user-name*) need read access to the group for which you want to display attribute and known members information.
- You also need test access to determine if a user is a member of a group.

SHOW KNOWN CLEARINGHOUSES

Displays information for all clearinghouses known to the local DNS server.

Format

SHOW KNOWN CLEARINGHOUSES

Required Access Rights

No specific access rights are needed. Requires TMPMBX and NETMBX privileges.

SHOW LINK

Displays all attribute names or the value of a specific attribute for a link. See Appendix B for attribute information.

Format

```
SHOW LINK   link-name { ATTRIBUTE att-name   }
                      { KNOWN ATTRIBUTES     }
```

Parameters

link-name
Name of the link whose attribute information you want to display.

att-name
Name of the link attribute for which you want information displayed..

Examples

1. `DNS> SHOW LINK SALES.LINKEY KNOWN ATTRIBUTES`

   This command displays the names of attributes for the link SALES.LINKEY.

2. `DNS> SHOW LINK SALES.LINKEY ATTRIBUTE DNS$LINKTARGET`

   This command displays the destination of the link SALES.LINKEY.

Required Access Rights

To use this command, you (expressed as *node-name*::*user-name*) need read access to the link.

SHOW NAMESERVER

Displays the following information about the namespace:

- Namespace name
- Nameserver state
- UID
- Timezone
- Incompatible request counter
- Time went backwards counter

Format

SHOW NAMESERVER

Required Access Rights

No specific access rights are needed. Requires TMPMBX privilege.

SHOW OBJECT

Displays the value of a specified attribute or the name of all known attributes for an object. See Appendix B for attribute information.

Format

$$\textbf{SHOW OBJECT}\quad \textit{obj-name}\left\{\begin{array}{l}\text{ATTRIBUTE }\textit{att-name}\\ \text{KNOWN ATTRIBUTES}\end{array}\right\}$$

Parameters

obj-name
Name of the object whose attribute information you want to display.

att-name
Name of the attribute whose value you want to display.

Examples

1.
```
DNS> SHOW OBJECT SALES.OFFICE.87 ATTRIBUTE DNS$CLASS
```

   This command displays the class of object SALES.OFFICE.87.

2.
```
DNS> SHOW SALES.OFFICE.87 KNOWN ATTRIBUTES
```

   This command displays the names of all attributes for object SALES.OFFICE.87.

Required Access Rights

To use this command, you (expressed as *node-name::user-name*) need read access to the object.

SHOW VERSION

Displays the current DNS software version and protocol version on this node.

Format

SHOW VERSION

Required Access Rights

You do not need specific access rights.

START CLEARINGHOUSE

Starts a clearinghouse.

Format

START CLEARINGHOUSE *ch-name*

Parameters

ch-name
Name of the clearinghouse you want to start.

Required Access Rights

- To use this command, you (expressed as *node-name*::*user-name*) need read and write access to the clearinghouse object and the clearinghouse entry.
- Requires system manager privileges.

STOP CLEARINGHOUSE

Stops a clearinghouse.

Format

STOP CLEARINGHOUSE *ch-name*

Parameters

ch-name
Name of the clearinghouse that you want to stop.

Required Access Rights

Requires system manager privileges.

UPDATE DIRECTORY

Forces an immediate update on a directory.

Format

UPDATE DIRECTORY *dir-name*

Parameters

dir-name
Name of the directory to be updated.

Examples

1.
```
DNS> UPDATE DIRECTORY A.B
```

   This command forces an update on directory A.B.

Required Access Rights

To use this command, you (expressed as *node-name::user-name*) need control access to the directory.

Appendix A
Error Handling

This appendix lists error messages issued by the VAX Distributed Name Service. It also describes the conditions that caused the message to be issued and how you can correct the problem.

Messages are listed in alphabetical order.

%DNS-E-ACCESSDENIED, No rights to perform requested operation

Explanation. The user does not have appropriate access rights to perform the requested operation.

User Action. Contact the network administrator to get the appropriate access to the entry.

%DNS-E-ALREADYREPLICA, The clearinghouse already contains a copy of the directory

Explanation. The user tried to place a directory copy in a clearinghouse that already contained a copy of that directory.

User Action. None.

%DNS-E-BADCLOCK, DNS server clocks not synchronized

Explanation. The Greenwich Mean Time difference among DNS server clocks is greater than four minutes.

User Action. Reset one or more DNS server clocks or timezones.

%DNS-E-BADEPOCH, Directory copies not synchronized

Explanation. The update procedure found that the DNS$Epoch attribute was not identical for copies of a directory being updated.

User Action. Issue the DNS$CONTROL REBUILD DIRECTORY command to bring the set of copies into conformity.

%DNS-E-CANTPUTHERE, The directory cannot be created in this clearinghouse

Explanation. A directory must be stored in at least one clearinghouse closer to the root than the directory itself. Closer to the root is defined as fewer levels (.) in the name.

User Action. Choose a clearinghouse closer to the root.

%DNS-E-CLEARINGHOUSEDOWN, Clearinghouse is not available

Explanation. A requested clearinghouse is present at the DNS server but is not available.

User Action. Start the clearinghouse by issuing the DNS$CONTROL START CLEARINGHOUSE command.

%DNS-E-CLERKBUG, Bug, submit an SPR

Explanation. A bug was found in the client interface.

User Action. Submit a Software Performance Report.

%DNS-E-CRUCIALREPLICA, The directory cannot be removed

Explanation. This copy of the directory cannot be removed without compromising the integrity of the namespace. Possible reasons are:

- The copy is needed so that the root can be reached..
- The copy is the master.

User Action. None.

%DNS-E-DATACORRUPTION, DNS server is unable to access clearinghouse datafile

Explanation. An error occurred while DNS was accessing data stored at a clearinghouse.

User Action. Issue the REBUILD CLEARINGHOUSE command or use a backup copy of the clearinghouse. Refer to the DECnet event log for more information on the cause of the error.

%DNS-E-ENTRYEXISTS, Creation was not possible,entry already exists.

Explanation. Creation of this entry was not possible because either an entry of the same name already exists or a directory of the same name is in the process of being deleted.

User Action. Reenter the command with another name.

%DNS-E-GBLSECERROR, Error accessing DNS$GLOBAL section

Explanation. The DNS server global section, SYS$SYSTEM:DNS$GLOBAL.GBI is not mapped. The DNS server was not started at this node.

User Action. Use DNS$CONTROL to start the DNS server.

%DNS-E-INVALIDNAME, Invalid name

Explanation. The name of an entry was badly formed.

User Action. Reenter the command with the correct syntax.

%DNS-E-MORETHANONEREPLICA, The directory has more than a single copy

Explanation. You tried to delete a directory that still has copies.

User Action. Issue the DNS$CONTROL SHOW DIRECTORY *dir_name* ATTRIBUTE DNS$REPLICAS command to display the clearinghouses that contain copies of the directory. Delete all copies before you try to delete the master.

%DNS-E-NAMESERVERBUG, Bug, submit an SPR

Explanation. A bug was found in the DNS server.

User Action. Submit a Software Performance Report.

%DNS-E-NOCOMMUNICATION, Unable to communicate with DNS server

Explanation. You cannot communicate with any DNS server capable of processing the request. It was impossible to establish a link, the DNS server was not operating, or a clearinghouse containing a a directory copy is stopped.

User Action. Make sure that the remote DNS server is running and verify DECnet connectivity to the remote DNS server node, or make sure that all clearinghouses containing a copy of the directory are on.

%DNS-E-NONSNAME

Explanation. You specified an unknown namespace.

User Action. Reenter the command using another namespace name.

%DNS-E-NONSRESOURCES, Insufficient resources at DNS server

Explanation. Resources were insufficient to process the request at the DNS server.

User Action. Verify that the DNS server has adequate resources.

%DNS-E-NOTAGROUP, Not a group

Explanation. You requested a group operation on an object whose class was not DNS$Group.

User Action. Reenter the command with a new group name.

%DNS-E-NOTAREPLICA, The clearinghouse does not contain a copy of the directory

Explanation. You tried to access a directory copy that did not exist in the specified clearinghouse.

User Action. Use the DNS$CONTROL SHOW DIRECTORY *dir_name* ATTRIBUTE DNS$REPLICAS command to display the names of the clearinghouses that contain copies of this directory.

%DNS-E-NOTCHDIRECTORY, The parent directory is not a clearinghouse directory

Explanation. You can store clearinghouses only in directories that are allowed to store clearinghouse objects (that is, the directory has the DNS$InCHName attribute set to 1).

User Action. To allow a directory to store clearinghouse objects, issue the DNS$CONTROL SET DIRECTORY *directory name* ALLOW command. Alternately, issue the ADD ACCESS command for the *node-name*::*user-name* and *node-name*::DNS$SERVER on the directory where you want to store the clearinghouse object.

%DNS-E-NOTEMPTY, Clearinghouse or directory is not empty

Explanation. An attempt was made to delete a clearinghouse or directory that was not empty.

User Action. Use DNS$CONTROL to remove all copies of directories.

%DNS-E-NOTLINKED, A link is not contained in the name

Explanation. The supplied name does not contain a link.

User Action. None.

%DNS-E-OLDSKULK, UPDATE procedure already in progress

Explanation. An old update procedure tried to complete after a new update procedure had already started.

User Action. None.

%DNS-E-POSSIBLECYCLE, Loop detected in link or group entry

Explanation. A possible loop was detected in a chain of links or groups. Two or more groups have each other as members.

User Action. Use DNS$CONTROL to follow the chain of links or groups. Remove the entry that is causing the loop.

%DNS-E-RESOURCEERROR, Insufficient resources to process request

Explanation. Resources are inadequate for processing the client's request.

User Action. Verify that the client has adequate resources.

%DNS-E-UNKNOWNCLEARINGHOUSE, The clearinghouse does not exist

Explanation. You tried to access a clearinghouse that does not exist.

User Action. Reenter the command with another clearinghouse name.

%DNS-E-UNKNOWNENTRY, Requested entry does not exist

Explanation. The requested entry does not exist, the user does not have access to the entry, or the entry is in the process of being deleted.

User Action. None.

%DNS-E-WRONGSTATE, DNS server is in the wrong state

Explanation. The DNS server or a clearinghouse is in the wrong state to perform this function. States are ON, OFF, DYING, and INITIALIZING.

User Action. Start or stop the DNS server or clearinghouse, as appropriate.

Appendix B
Attribute Tables

This appendix contains two tables. Table B-1 identifies and defines DNS global attributes. Table B-2 lists the global attributes defined by DNS, specifies whether they are single-value or set attributes, tells whether they can be modified, and identifies the kind of entry to which each applies.

Table B-1: Global Attributes

Attribute	Definition
For all entries:	
DNS$UID	The unique identifier. A unique numeric value assigned by DNS when a clearinghouse, directory, group, link, or object is created.
DNS$UTS	Update timestamp. The timestamp for the most recent update to any attribute.
DNS$ACS	Access control set. A set of individuals or groups that have access to an entry.
For object entries:	
DNS$Class	Specifies the class to which an object belongs.
DNS$ClassVersion	The version number of this class. Allows applications to build in compatibility with entries created by earlier versions.

(continued on next page)

Table B-1 (Cont.): Global Attributes

Attribute	Definition
DNS$Address	Addresses of nodes where an object can be found.
DNS$ObjectUID	Application-specific attribute that is created and maintained by the application.
For directory entries:	
DNS$Replicas	The address, UID, and name of a set of clearinghouses where a copy of this directory is located. Specifies whether the directory in that clearinghouse is a master or a read-only copy.
DNS$AllUpTo	Specifies the last successful update.
DNS$Convergence	Specifies how often the propagate mechanism should update all read-only copies of a master directory. A value of LOW postpones the update. A value of HIGH makes one attempt to propagate changes; if unsuccessful, the update is tried again at the next scheduled update. Controlled by the SET DIRECTORY CONVERGENCE command.
DNS$InCHName	A directory attribute that indicates whether this directory can store clearinghouse objects. Controlled by the SET DIRECTORY ALLOW (or DISALLOW) command.
DNS$ReplicaState	Used by DNS software. Specifies whether a directory can be accessed.
DNS$ReplicaType	Indicates whether a directory is a master or a read-only copy.
DNS$LastSkulk	Used by DNS software. Timestamp of the last update performed on this directory. Ensures that multiple, concurrent updates do not conflict. Guarantees that modifications made during updates are not missed.

(continued on next page)

Table B-1 (Cont.): Global Attributes

Attribute	Definition
DNS$LastUpdate	Used by DNS software. Timestamp of last update to any directory attribute or any change to directory content, including object entries, child pointer entries, and link entries. Controls propagation of updates to other copies of the directory and limits the number of updates needed.
DNS$RingPointer	Used by DNS software. Exists only on master directories. A pointer to the clearinghouse containing the next copy of this directory in a virtual ring of copies. Coordinates the convergence of distributed updates during an update. Contains the UID of the next clearinghouse in the ring.
DNS$Epoch	Identifies the particular incarnation of the directory. Epochs are defined the REBUILD DIRECTORY command. All copies of a directory must have the same epoch number in order for updates for that directory to succeed.
DNS$ParentPointer	Contains the name of the parent directory. Also specifies the expiration date and extension time of the pointer.
For link entries:	
DNS$LinkTarget	Full name of the entry to which a link points.
DNS$LinkTimeout	Timeout data specifying when the target of a link should be checked (value is greater than zero). If value is zero, the link is permanent.
For child pointers:	
DNS$ChildUID	A copy of the DNS$UID of the directory entry for the child pointer.
DNS$Replicas	Same as for directories.

(continued on next page)

Table B-1 (Cont.): Global Attributes

Attribute	Definition
For clearinghouse entries:	
DNS$CHupPointer	An attribute set. Each member is a pointer to a clearinghouse that stores a copy of at least one directory that is closer to the root directory than any directory stored in this clearinghouse. If this clearinghouse contains a copy of the root directory, then this attribute is null or absent.
DNS$CHName	Name of this clearinghouse.
DNS$CHLastAddress	Network address at which the clearinghouse last reported itself.
DNS$CHState	Specifies whether the clearinghouse can respond to requests regarding the directories it stores.
DNS$CHDirectories	Attribute set that contains one member for each directory copy stored in the clearinghouse.
DNS$NSNickname	Stores the name of the namespace of which it is a part.
DNS$NSUID	Stores the UID of the namespace of which it is a part.
For DNS-reserved objects:	
DNS$Group	Indicates that the object is a group with an additional attribute called DNS$Members. Grants access rights to DNS entries.
DNS$Clearinghouse	Indicates a special object called the clearinghouse object.

Table B-2 summarizes certain characteristics of the global attributes.

Table B-2: Summary of Global Attributes

Attribute Name	Single/Set	Action	Applies to
DNS$UID	Single	Read	All
DNS$UTS	Single	Read	All
DNS$ACS	Set	Read/write	All
DNS$ChildUID	Single	Read	Child
DNS$Class	Single	Read	Object
DNS$ClassVersion	Single	Read/write	Object
DNS$Address	Set	Read/write/delete	Object,clearinghouse
DNS$ObjectUID	Single	Read/write	Object
DNS$Replicas	Set	Read	Directory,child
DNS$AllUpto	Single	Read	Directory
DNS$Convergence	Single	Read/write	Directory
DNS$InCHName	Single	Read/write/delete	Directory
DNS$ParentPointer	Set	Read	Directory
DNS$ReplicaState	Single	Read	Directory
DNS$ReplicaType	Single	Read	Directory
DNS$LastSkulk	Single	Read	Directory
DNS$LastUpdate	Single	Read	Directory
DNS$RingPointer	Single	Read	Directory
DNS$Epoch	Single	Read	Directory
DNS$LinkTarget	Single	Read/write	Link
DNS$LinkTimeout	Single	Read/write	Link
DNS$CHupPointers	Set	Read	Clearinghouse
DNS$CHname	Set	Read	Clearinghouse
DNS$CHLastAddress	Single	Read	Clearinghouse
DNS$CHState	Single	Read	Clearinghouse
DNS$CHDirectories	Set	Read	Clearinghouse
DNS$NSNickname	Single	Read	Clearinghouse

Appendix C

DNS Events

DNS uses the DECnet event logger (EVL) to log significant events. As described in Chapter 3, event information can help you maintain the network because it can be recorded continuously by the EVL.

This appendix lists DNS event messages, gives the meaning of each, and, where appropriate, tells you what action to take.

C.1 Event Messages

Table C-1 lists DNS events and the code number and description of each. Information that DNS supplies appears in braces ({ }).

Table C-1: DNS Events

Event Name	Code	Message Logged
IncompatibleRequest	352.1	Request received with an unsupported value of MajorVersion {version received, source, principal}.
Security	352.3	DNS server encountered a security violation attempting to access name {name being accessed, source, principal}.
TreeBroken	352.4	The namespace appears disconnected; an ancestor of directory {name being accessed} does not exist.
RootLost	352.5	The root cannot be found starting at clearinghouse {clearinghouse name}.
CrucialReplica	352.7	Two RemoveReplica operations together would violate the crucial replica restraints. The attempt to remove the replica of directory {directory name} at clearinghouse {clearinghouse name} has been canceled and the replica returned to the ON state.
ParentPointer	352.8	Unable to update child pointer for directory {directory name}. Some copies of this directory may be temporarily unreachable.
CHCreated	352.9	Clearinghouse {clearinghouse name} created.
DataCorruption	352.10	Clearinghouse {clearinghouse name} may have corrupted data for the following reason: {implementation-dependent reason}.
PossibleCycle	352.11	Possible loop detected while resolving name {name}.
SkulkFailed	352.12	An update procedure on directory {directory name} running at clearinghouse {clearinghouse name} failed because {condition causing UPDATE to fail}.

(continued on next page)

Table C-1 (Cont.): DNS Events

Event Name	Code	Message Logged
CHEntryGone	352.13	Clearinghouse entry does not exist or is missing.
CHEnabled	352.14	Clearinghouse started.
CHDisabled	352.15	Clearinghouse stopped.
CHDeleted	352.16	Clearinghouse deleted.
RMS Event	353.1	{RMS error message}
Background complete event	353.2	Background started: {time} completed: {time}
PropagateFailed	353.3	A propagate procedure on directory {directory name} running at clearinghouse {clearinghouse name} failed because {condition causing PROPAGATE to fail}
Backwards time event	353.4	The system time has gone backward. The system clock must be set after {time}.

C.2 Meaning of Event Messages and Action to Take

Event Code 352.1: The version of DNS software on this server does not support the client request.

Action: Submit a Software Performance Report (SPR).

Event Code 352.3: A request was received from another DNS server that did not have proper access to the clearinghouse.

Action: Issue the DNS$CONTROL ADD ACCESS command to grant access to the clearinghouse.

Event Code 352.4: A child pointer could not be accessed from this DNS server. Access to the child pointer is controlled by the parent directory.

Action: Make sure that the DNS server has read, write, and delete access to the parent directory.

Event Code 352.5: A copy of a directory closer to the root was not found in this clearinghouse.

Action: Issue the DNS$CONTROL COPY DIRECTORY command to place a copy of the directory in this clearinghouse. If this condition is not corrected, the DNS server cannot guarantee that any lookup procedure, initiated at any clearinghouse, will be able to find the root.

Event Code 352.7: You tried to remove a copy of the same directory from more than one clearinghouse. While each action by itself is valid, the combination is not. Each directory must have at least one copy stored in a clearinghouse closer to the root than the directory itself. If this rule is not followed, it may be impossible to locate the directory.

Action: One copy of the crucial replica that was marked for delete was turned back on. If you want to remove it from the clearinghouse, copy it to a clearinghouse closer to the root.

Event Code 352.8: The set of copies of this directory may not match the child pointer. Some copies may be temporarily unreachable.

Action: If event 352.4 was logged with this one, take action on that event first. If this event was logged without event 352.4, it means that one of the clearinghouses could not be contacted. The condition will be corrected when communication is restored.

Event Code 352.10: An error occurred while DNS was accessing the clearinghouse file.

Action: First check RMS error messages and take appropriate action. If that does not help, copy the clearinghouse to another area of the disk. If that does not resolve the problem, restore a backup copy and issue the DNS$CONTROL REBUILD DIRECTORY command for all the directories in that clearinghouse. Updates not yet propagated or recorded on the backup copy are lost.

Event Code 352.11: A loop was detected while DNS was resolving a name.

Action: Issue the DNS$CONTROL SHOW LINK *link-name* ATTRIBUTE DNS$LinkTarget command recursively, searching for a loop.

Event Code 352.12: The update failed while trying to contact the clearinghouse, due to the reason specified in the event.

Action: Specific reasons for the update failure are listed. Check the DNS error messages in Appendix A and take the indicated action.

Event Code 353.1: Error occurred accessing the file. RMS error text displayed. This event occurs most commonly when there is not enough disk space to create or expand the clearinghouse.

Action: Purge files to gain 4200 disk blocks.

Event Code 353.3: The propagate procedure failed while trying to contact the clearinghouse, due to the reason specified in the event.

Action: Specific reasons for the propagate failure are listed. Check the DNS error messages in Appendix A and take the indicated action.

Event Code 353.4: The system time has gone backward.

Action: Reset the system clock.

Appendix D

Valid Characters for DNS Names

A DNS name consists of a string of up to 255 characters in length. The following variants are allowed:

- Any character in Tables D-1 and D-2.
- Binary name: Names containing codes must be prefixed with %X or %x. The binary name itself must consist of pairs of hexadecimal digits (for example, 0–9, A–F, a–f).
- Wildcard name: You can specify the asterisk (*) and the question mark (?) in a DNS name to indicate wildcarding. The question mark matches one character in the corresponding name. The asterisk matches any number of characters in the corresponding name.

Table D-1 lists the characters that you can include in unquoted DNS names, that is, DNS names that you do not enclose in quotes (" "). If your DNS names contains any of the characters in Table D-2, you must enclose the name in quotes (" ").

Table D-1: Valid Character Codes in Unquoted DNS Names

Graphic	Decimal Value	Abbreviation	Description
$	36	$	dollar sign
-	45	-	hyphen or minus
0	48	0	zero
1	49	1	one
2	50	2	two
3	51	3	three
4	52	4	four
5	53	5	five
6	54	6	six
7	55	7	seven
8	56	8	eight
9	57	9	nine
A	65	A	uppercase A
B	66	B	uppercase B
C	67	C	uppercase C
D	68	D	uppercase D
E	69	E	uppercase E
F	70	F	uppercase F
G	71	G	uppercase G
H	72	H	uppercase H
I	73	I	uppercase I
J	74	J	uppercase J
K	75	K	uppercase K
L	76	L	uppercase L
M	77	M	uppercase M
N	78	N	uppercase N
O	79	O	uppercase O
P	80	P	uppercase P
Q	81	Q	uppercase Q

(continued on next page)

Table D-1 (Cont.): Valid Character Codes in Unquoted DNS Names

Graphic	Decimal Value	Abbreviation	Description
R	82	R	uppercase R
S	83	S	uppercase S
T	84	T	uppercase T
U	85	U	uppercase U
V	86	V	uppercase V
W	87	W	uppercase W
X	88	X	uppercase X
Y	89	Y	uppercase Y
Z	90	Z	uppercase Z
_	95	_	underline (underscore)
a	97	a	lowercase a
b	98	b	lowercase b
c	99	c	lowercase c
d	100	d	lowercase d
e	101	e	lowercase e
f	102	f	lowercase f
g	103	g	lowercase g
h	104	h	lowercase h
i	105	i	lowercase i
j	106	j	lowercase j
k	107	k	lowercase k
l	108	l	lowercase l
m	109	m	lowercase m
n	110	n	lowercase n
o	111	o	lowercase o
p	112	p	lowercase p
q	113	q	lowercase q
r	114	r	lowercase r
s	115	s	lowercase s

(continued on next page)

Table D-1 (Cont.): Valid Character Codes in Unquoted DNS Names

Graphic	Decimal Value	Abbreviation	Description
t	116	t	lowercase t
u	117	u	lowercase u
v	118	v	lowercase v
w	119	w	lowercase w
x	120	x	lowercase x
y	121	y	lowercase y
z	122	z	lowercase z
À	192	À	uppercase A with grave accent
Á	193	Á	uppercase A with acute accent
Â	194	Â	uppercase A with circumflex
Ã	195	Ã	uppercase A with tilde
Ä	196	Ä	uppercase A with umlaut (dieresis)
Å	197	Å	uppercase A with ring
Æ	198	Æ	uppercase AE diphthong
Ç	199	Ç	uppercase C with cedilla
È	200	È	uppercase E with grave accent
É	201	É	uppercase E with acute accent
Ê	202	Ê	uppercase E with circumflex
Ë	203	Ë	uppercase E with umlaut (dieresis)
Ì	204	Ì	uppercase I with grave accent
Í	205	Í	uppercase I with acute accent
Î	206	Î	uppercase I with circumflex
Ï	207	Ï	uppercase I with umlaut (dieresis)
Ñ	209	Ñ	uppercase N with tilde
Ò	210	Ò	uppercase O with grave accent
Ó	211	Ó	uppercase O with acute accent
Ô	212	Ô	uppercase O with circumflex
Õ	213	Õ	uppercase O with tilde
Ö	214	Ö	uppercase O with umlaut (dieresis)

(continued on next page)

Table D-1 (Cont.): Valid Character Codes in Unquoted DNS Names

Graphic	Decimal Value	Abbreviation	Description
Ø	216	Ø	uppercase O with slash
Ù	217	Ù	uppercase U with grave accent
Ú	218	Ú	uppercase U with acute accent
Û	219	Û	uppercase U with circumflex
Ü	220	Ü	uppercase U with umlaut (dieresis)
ß	223	ß	German lowercase sharp s
à	224	à	lowercase a with grave accent
á	225	á	lowercase a with acute accent
â	226	â	lowercase a with circumflex
ã	227	ã	lowercase a with tilde
ä	228	ä	lowercase a with umlaut (dieresis)
å	229	å	lowercase a with ring
æ	230	æ	lowercase ae diphthong
ç	231	ç	lowercase c with cedilla
è	232	è	lowercase e with grave accent
é	233	é	lowercase e with acute accent
ê	234	ê	lowercase e with circumflex
ë	235	ë	lowercase e with umlaut (dieresis)
ì	236	ì	lowercase i with grave accent
í	237	í	lowercase i with acute accent
î	238	î	lowercase i with circumflex
ï	239	ï	lowercase i with umlaut (dieresis)
ñ	241	ñ	lowercase n with tilde
ò	242	ò	lowercase o with grave accent
ó	243	ó	lowercase o with acute accent
ô	244	ô	lowercase o with circumflex
õ	245	õ	lowercase o with tilde
ö	246	ö	lowercase o with umlaut (dieresis)
ø	248	ø	lowercase o with slash

(continued on next page)

Table D-1 (Cont.): Valid Character Codes in Unquoted DNS Names

Graphic	Decimal Value	Abbreviation	Description
ù	249	ù	lowercase u with grave accent
ú	250	ú	lowercase u with acute accent
û	251	û	lowercase u with circumflex
ü	252	ü	lowercase u with umlaut (dieresis)

Table D-2: Valid Character Codes in Quoted DNS Names

Graphic	Decimal Value	Abbreviation	Description
?	32	SP	space
!	33	!	exclamation point
#	35	#	number sign
$	36	$	dollar sign
%	37	%	percent sign
&	38	&	ampersand
'	39	'	apostrophe (single quote)
(	40	(	opening parenthesis
)	41	)	closing parenthesis
*	42	*	asterisk
+	43	+	plus sign
,	44	,	comma
.	46	.	period or decimal point
/	47	/	slash
:	58	:	colon
;	59	;	semicolon
<	60	<	less-than sign
=	61	=	equals sign
>	62	>	greater-than sign
?	63	?	question mark
@	64	@	commercial "at" sign
[	91	[	opening bracket
\	92	\	back slash
]	93	]	closing bracket
^	94	^	circumflex
`	96	`	grave accent
{	123	{	opening brace
|	124	|	vertical line
}	125	}	closing brace

(continued on next page)

Table D-2 (Cont.): Valid Character Codes in Quoted DNS Names

Graphic	Decimal Value	Abbreviation	Description
~	126	~	tilde
¡	161	¡	inverted exclamation mark
¢	162	¢	cent sign
£	163	£	pound sign
¥	165	¥	yen sign
§	167	§	section sign
©	169	©	copyright sign
ª	170	ª	feminine ordinal indicator
«	171	«	angle quotation mark, left
°	176	°	degree sign
±	177	±	plus or minus sign
^{2}	178	^{2}	superscript 2
^{3}	179	^{3}	superscript 3
µ	181	µ	micro sign
¶	182	¶	paragraph sign, pilcrow
·	183	·	middle dot
^{1}	185	^{1}	superscript 1
º	186	º	masculine ordinal indicator
»	187	»	angle quotation mark, right
¼	188	¼	fraction, one quarter
½	189	½	fraction, one half
¿	191	¿	inverted question mark

Glossary

access right

A privilege that enables a user to read or manipulate a clearinghouse, directory, object, or link entry. There are five types of access: read, write, delete, control, and test. The DNS installation procedure grants all rights to all users. Anyone with control access to an entry can change or remove rights. All rights automatically belong to the creator of the entry, who can grant any access rights to other users.

access control set (ACS)

A group of attributes that determines whether or not access should be granted. Each clearinghouse, directory, link, and object entry has an associated ACS.

attribute

A property of an entry. For example, an object's network address is one of its attributes. DNS assigns global attributes to clearinghouses, directories, objects, and links. Applications can assign other attributes to the objects they create.

attribute name

A name that identifies an attribute. The names of global attributes have the prefix DNS$ and are defined by DNS.

attribute set

A collection of values that can apply to a single attribute.

attribute value

A single value of an attribute.

child pointer

An entry in a directory that points to a related directory. For example, directory A.B.C is a child of directory A.B. The child pointer is stored in the parent directory. It is automatically created when the child directory is created and is deleted when the child directory is deleted.

class

Refers to DNS$Class, which is a mandatory attribute of an object entry. This attribute identifies objects as belonging to a specific application.

class-specific attribute

An attribute that is defined by an application, not by DNS.

clearinghouse

A collection of one or more directories on a DNS server. A clearinghouse is created on a node when a DNS server is installed on that node.

client

A DFS or RSM node that contains DNS client software. The client communicates with the DNS server to store and receive object name information.

DNS server

A node on which DNS software has been installed. Object names are stored on DNS servers.

directory

A named entity that contains three kinds of entries: object entries, child pointer entries, and link entries. DNS directories are stored in clearinghouses.

DNS$CONTROL

The VAX Distributed Name Service Control Program. You issue DNS management commands through this utility.

global attribute

An attribute whose meaning is the same for all entries stored by DNS. DNS defines global attributes.

group

An object with the class attribute DNS$Group. You create a group so that you can grant its members common access rights. Group members can be individual users or other groups.

hierarchical namespace

A namespace that contains a hierarchy of DNS directories.

link

An entry in the namespace that points to another name.

master directory

A directory that can be updated. You create a master directory with the DNS$CONTROL CREATE DIRECTORY or REBUILD DIRECTORY command.

namespace

The logical entity within which DNS operates. You create the namespace by placing object, directory, and link entries in it.

object

A network resource that, in DNS, has a name and a set of attributes.

object entry

The name and set of attributes that represent a object in the namespace.

read-only directory

A copy of a directory that is used for name lookup purposes only. DNS updates a read-only directory by updating its master directory.

replica

Another name for a copy of a directory.

replica type

A directory attribute that indicates whether a copy of a directory is a master or a read-only copy.

root directory

The directory created by installation of DNS software. It is unnamed and is represented by a dot (.).

single-directory namespace

A namespace that contains only one directory; the root directory.

skulk

Another name for an update procedure. An update procedure propagates changes to a master directory to all read-only copies of the master directory.

update

A procedure that propagates changes to a master directory to all read-only copies of the master directory. The update procedure runs automatically approximately every 12 or 24 hours, depending on the value you specify for the DNS$Convergence attribute. You can run the update procedure at any time.

UID (unique identifier)

A value that identifies the namespace and its entries. The UID consists of time and network address information. When the namespace or any of its entries are successfully created, the UID is displayed.

user

A person or an application that uses the name service.

wildcard name

A name that allows you to look up groups of entries that have common name characteristics. An asterisk (*) alone displays all names; an asterisk accompanied by some name information displays all names that include the specified information. A question mark (?) accompanied by some name information displays names that have the specified characteristics plus one more in place of the question mark.

Index

HOW TO ORDER ADDITIONAL DOCUMENTATION

DIRECT TELEPHONE ORDERS

In Continental USA
call 800–DIGITAL

In Puerto Rico
call 809–754–7575 x2012

In Canada
call 800–267–6215

In New Hampshire
Alaska or Hawaii
call 603–884–6660

ELECTRONIC ORDERS (U.S. ONLY)

Dial 800–DEC–DEMO with any VT100 or VT200
compatible terminal and a 1200 baud modem.
If you need assistance, call 1–800–DIGITAL.

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.
940 Belfast Road
Ottawa, Ontario, Canada K1G 4C2
Attn: A&SG Business Manager

INTERNATIONAL

DIGITAL
EQUIPMENT CORPORATION
A&SG 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 x2012

VAX Distributed Name Service
Management Guide
AA–KN85B–TE

READER'S COMMENTS

What do you think of this manual? Your comments and suggestions will help us to improve the quality and usefulness of our publications.

Please rate this manual:

	Poor				Excellent
Accuracy	1	2	3	4	5
Readability	1	2	3	4	5
Examples	1	2	3	4	5
Organization	1	2	3	4	5
Completeness	1	2	3	4	5

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

General comments:

Suggestions for improvement:

Name ______________________ Date ______________________

Title ______________________ Department ______________________

Company ______________________ Street ______________________

City ______________ State/Country ______________ Zip Code ______________