Compaq TCP/IP Services for OpenVMS

Release Notes

April 2002

This document describes the new features and changes to the Compaq TCP/IP Services for OpenVMS Version 5.3 software product.

Revision/Update Information:	These release notes supersede the Compaq TCP/IP Services for OpenVMS V5.1 Release Notes.
Software Version:	Compaq TCP/IP Services for OpenVMS Version 5.3
Operating Systems:	OpenVMS Alpha Versions 7.2-2, 7.3 OpenVMS VAX Versions 7.2, 7.3

Compaq Computer Corporation
Houston, Texas

Contents

2 Installation, Configuration, and Startup Notes

3 Problems and Restrictions

B Configuring and Managing NTP

C Configuring and Managing BIND Version 9

D Advanced IPv6 Programming Socket Interface

Tables

Preface

The Compaq TCP/IP Services for OpenVMS product is the Compaq implementation of the TCP/IP protocol suite and internet services for OpenVMS Alpha and OpenVMS VAX systems. This document describes the Compaq TCP/IP Services for OpenVMS Version 5.3 product.

TCP/IP Services provides a comprehensive suite of functions and applications that support industry-standard protocols for heterogeneous network communications and resource sharing.

For installation instructions, see the *Compaq TCP/IP Services for OpenVMS Installation and Configuration* manual.

The release notes provide version-specific information that supersedes the information in the documentation set. The features, restrictions, and corrections in this version of the software are described in the release notes. Always read the release notes before installing the software.

Intended Audience

These release notes are intended for experienced OpenVMS and UNIX system managers and assumes a working knowledge of OpenVMS system management, TCP/IP networking, TCP/IP terminology, and some familiarity with the TCP/IP Services product.

Related Documents

Table 1 lists the documents available with this version of TCP/IP Services.

Table 1 TCP/IP Services Documentation

Manual	Contents
Compaq TCP/IP Services for OpenVMS Concepts and Planning	This manual provides conceptual information about TCP/IP networking on OpenVMS systems, including general planning issues to consider before configuring your system to use the TCP/IP Services software. This manual also describes the other manuals in the TCP/IP Services documentation set and provides a glossary of terms and acronyms for the TCP/IP Services software product.
Compaq TCP/IP Services for OpenVMS Release Notes	The release notes provide version-specific information that supersedes the information in the documentation set. The features, restrictions, and corrections in this version of the software are described in the release notes. Always read the release notes before installing the software.

(continued on next page)

Table 1 (Cont.) TCP/IP Services Documentation

Manual	Contents
Compaq TCP/IP Services for OpenVMS Installation and Configuration	This manual explains how to install and configure the TCP/IP Services product.
Compaq TCP/IP Services for OpenVMS User's Guide	This manual describes how to use the applications available with TCP/IP Services such as remote file operations, e-mail, TELNET, TN3270, and network printing. This manual explains how to use these services to communicate with systems on private internets or on the worldwide Internet.
Compaq TCP/IP Services for OpenVMS Management	This manual describes how to configure and manage the TCP/IP Services product. Use this manual with the *Compaq TCP/IP Services for OpenVMS Management Command Reference* manual.
Compaq TCP/IP Services for OpenVMS Management Command Reference	This manual describes the TCP/IP Services management commands. Use this manual with the *Compaq TCP/IP Services for OpenVMS Management* manual.
Compaq TCP/IP Services for OpenVMS Management Command Quick Reference Card	This reference card lists the TCP/IP management commands by component and describes the purpose of each command.
Compaq TCP/IP Services for OpenVMS UNIX Command Reference Card	This reference card contains information about commonly performed network management tasks and their corresponding TCP/IP management and Compaq *Tru64* UNIX command formats.
Compaq TCP/IP Services for OpenVMS ONC RPC Programming	This manual presents an overview of high-level programming using open network computing remote procedure calls (ONC RPC). This manual also describes the RPC programming interface and how to use the RPCGEN protocol compiler to create applications.
Compaq TCP/IP Services for OpenVMS Sockets API and System Services Programming	This manual describes how to use the Sockets API and OpenVMS system services to develop network applications.
Compaq TCP/IP Services for OpenVMS SNMP Programming and Reference	This manual describes the Simple Network Management Protocol (SNMP) and the SNMP application programming interface (eSNMP). It describes the subagents provided with TCP/IP Services, utilities provided for managing subagents, and how to build your own subagents.
Compaq TCP/IP Services for OpenVMS Tuning and Troubleshooting	This manual provides information about how to isolate the causes of network problems and how to tune the TCP/IP Services software for the best performance.
Compaq TCP/IP Services for OpenVMS Guide to IPv6	This manual describes the IPv6 environment, the roles of systems in this environment, the types and function of the different IPv6 addresses, and how to configure TCP/IP Services to access the 6bone network.

For additional information about Compaq *OpenVMS* products and services, access the Compaq website at the following location:

`http://www.compaq.com/openvms`

For a comprehensive overview of the TCP/IP protocol suite, you might find the book *Internetworking with TCP/IP: Principles, Protocols, and Architecture*, by Douglas Comer, useful.

Reader's Comments

Compaq welcomes your comments on this manual. Please send comments to either of the following addresses:

Internet	**openvmsdoc@compaq.com**
Mail	Compaq Computer Corporation OSSG Documentation Group, ZKO3-4/U08 110 Spit Brook Rd. Nashua, NH 03062-2698

How to Order Additional Documentation

Visit the following World Wide Web address for information about how to order additional documentation:

`http://www.openvms.compaq.com/`

Conventions

In the product documentation, the name TCP/IP Services means both:

- Compaq TCP/IP Services for OpenVMS Alpha
- Compaq TCP/IP Services for OpenVMS VAX

The name UNIX refers to the Compaq *Tru64* UNIX operating system.

The following conventions are used in the documentation. In addition, please note that all IP addresses are fictitious.

Ctrl/*x*	A sequence such as Ctrl/*x* indicates that you must hold down the key labeled Ctrl while you press another key or a pointing device button.
PF1 *x*	A sequence such as PF1 *x* indicates that you must first press and release the key labeled PF1 and then press and release another key or a pointing device button.
Return	In examples, a key name enclosed in a box indicates that you press a key on the keyboard. (In text, a key name is not enclosed in a box.) In the HTML version of this document, this convention appears as brackets, rather than a box.
. . .	A horizontal ellipsis in examples indicates one of the following possibilities: • Additional optional arguments in a statement have been omitted. • The preceding item or items can be repeated one or more times. • Additional parameters, values, or other information can be entered.
. . .	A vertical ellipsis indicates the omission of items from a code example or command format; the items are omitted because they are not important to the topic being discussed.

( )	In command format descriptions, parentheses indicate that you must enclose choices in parentheses if you specify more than one.
[ ]	In command format descriptions, brackets indicate optional choices. You can choose one or more items or no items. Do not type the brackets on the command line. However, you must include the brackets in the syntax for OpenVMS directory specifications and for a substring specification in an assignment statement.
|	In command format descriptions, vertical bars separate choices within brackets or braces. Within brackets, the choices are optional; within braces, at least one choice is required. Do not type the vertical bars on the command line.
{ }	In command format descriptions, braces indicate required choices; you must choose at least one of the items listed. Do not type the braces on the command line.
bold text	This typeface represents the introduction of a new term. It also represents the name of an argument, an attribute, or a reason.
italic text	Italic text indicates important information, complete titles of manuals, or variables. Variables include information that varies in system output (Internal error *number*), in command lines (/PRODUCER=*name*), and in command parameters in text (where *dd* represents the predefined code for the device type).
UPPERCASE TEXT	Uppercase text indicates a command, the name of a routine, the name of a file, or the abbreviation for a system privilege.
`Monospace text`	Monospace type indicates code examples and interactive screen displays. In the C programming language, monospace type in text identifies the following elements: keywords, the names of independently compiled external functions and files, syntax summaries, and references to variables or identifiers introduced in an example.
-	A hyphen at the end of a command format description, command line, or code line indicates that the command or statement continues on the following line.
numbers	All numbers in text are assumed to be decimal unless otherwise noted. Nondecimal radixes—binary, octal, or hexadecimal—are explicitly indicated.

1
New Features and Changes

This chapter describes the new features of Compaq TCP/IP Services for OpenVMS Version 5.3. For more information about configuring and managing these services, see the *Compaq TCP/IP Services for OpenVMS Management* guide provided with the TCP/IP Services for OpenVMS software.

Note

TCP/IP Services V5.3 is a direct upgrade from Version 5.1. Version 5.2 is a limited release.

Table 1–1 lists the new features of TCP/IP Services Version 5.3 and the sections that describe them.

Table 1–1 TCP/IP for OpenVMS Version 5.3 Features

Feature	Section
New Kernel Code Base	Section 1.1
NTP Version 4	Section 1.2
BIND Version 9	Section 1.3
IMAP	Section 1.4
Kerberos for TELNET	Section 1.5
SYSCONFIGTAB	Section 1.6
HELP for Startup and Shutdown Messages	Section 1.7
LPD Cluster Support	Section 1.8

1.1 New Kernel Code Base

Basic IPv6 support was provided in TCP/IP Services Version 5.1. Version 5.3 builds upon the basic IPv6 functionality to provide extended services, such as:

- IPv6 tunneling (as described in RFC 2473)
- 6-to-4 tunneling (no relay router)
- Anycast address
- Updated application programming interface (API) that conforms to the latest specification
- Mobile IPv6 correspondent node support (with processing of the binding update suboption for route optimization)

- Mobile IPv6 support in UNIX management tools (to display and decode Mobile IPv6 information)

The IPv6 enhancements are described in the following sections.

1.1.1 IPv6 Transition Mechanism

This release includes support for a new IPv6 transition mechanism called 6to4, as defined in RFC 3056.

In this release of TCP/IP Services, an OpenVMS system can operate either as a host in a 6to4 site or as a 6to4 border router (including support for being an isolated-host border router). Support for the 6to4 relay router is not currently provided.

An OpenVMS node can operate as a host within a 6to4 site without any explicit 6to4 configuration. The node obtains a 6to4 prefix using the standard IPv6 address autoconfiguration mechanisms. (That is, the 6to4 border router operating at the edge of the 6to4 site advertises the 6to4 prefix from which the host can automatically configure a 6to4 address.) Then, using the normal routing mechanisms, packets sent from the node to a 6to4 destination are directed to the border router.

Restrictions

The suggested rules for address selection that are outlined in RFC 3056 are not currently implemented. This might cause a node operating in a mixed 6to4 and native IPv6 site to incorrectly select the node's source address when sending IPv6 packets. Compaq recommends that you do not operate a host in a mixed 6to4 and native IPv6 site.

The OpenVMS border router will have exactly one interface to the IPv4-only cloud over which 6to4 traffic can be sent and received, and will have one or more IPv6 interfaces to the 6to4 site. (The IPv6 interfaces are attachments to different LANs in the same 6to4 site.)

The IETF has not defined how to handle issues of multihomed networks in IPv6. Therefore, Compaq recommends against setting up a multihomed IPv6 network at this time. When operating as a border router in a 6to4 site, an OpenVMS system should be the only border router in that site. Configuring multiple border routers within a 6to4 site is not recommended.

1.1.2 Advanced Programming Socket Interface

The version of TCP/IP Services supports the advanced sockets API for IPv6, as defined in `draft-ietf-ipngwg-rfc2292bis-06.txt`.

Note

The advanced sockets API for IPv6 defined in `draft-ietf-ipngwg-rfc2292bis-06.txt` is different from the advanced sockets API for IPv6 defined in RFC 2292. Any application that was coded for RFC 2292 will need to be updated to reflect the new advanced API.

For information about programming with the advanced sockets API, refer to Appendix D.

1.1.3 Mobile IPv6

This release of TCP/IP Services enables an OpenVMS node to operate as a mobile IPv6 correspondent node, as defined in the Internet draft *Mobility Support in IPv6* (`draft-ietf-mobileip-ipv6-15.txt`).

Note

Because this implementation is based on an IETF (Internet Engineering Task Force) draft, it is subject to change in future versions of TCP/IP Services.

This implementation does not support binding update authentication as specified in `draft-ietf-mobileip-ipv6-15.TXT`, Section 4.4, including the authentication data sub-option defined in Section 5.6. You should limit the use of this kit to test environments that are not subject to attack, since system integrity might be compromised by accepting unauthenticated bindings.

In a mobile IPv6 environment, nodes can have the following roles:

- Mobile node — a node (host or router) that can change its point of attachment from one link to another and still be reachable through its home address.
- Correspondent node — a peer node with which a mobile node is communicating. The correspondent node (host or router) can be either mobile or stationary.
- Home agent — a router on a mobile node's home link with which the mobile node registers its current care-of address. (Currently, OpenVMS cannot operate as a home agent).

IPv6 is designed to support mobility through its extensible header structure, address autoconfiguration, security (IPsec), and tunneling.

A node has a home address, which does not change; the node is always addressable by its home address. When a mobile node is on its home link, it is considered to be "at home." Packets destined for the mobile node's home address are delivered through standard IP routing mechanisms. When a mobile node moves to a foreign link, it is considered to be "away from home."

On the foreign link, the mobile node configures a care-of address and registers this new binding with its home agent by sending the home agent a binding update. This new address is the mobile node's primary care-of address. The home agent acknowledges the binding update by returning a binding acknowledgment to the mobile node.

Packets sent by a correspondent node to the mobile node's home address arrive at its home link. The home agent intercepts the packets, encapsulates them, and tunnels them to the mobile node's registered care-of address.

The mobile node receives the packets tunnelled to it from its home agent and recognizes its primary care-of address in the tunnelled packet's header. The mobile node assumes that the original sending correspondent node has no binding cache entry for the mobile node; otherwise, the correspondent node would have sent the packet directly to the mobile node using a routing header. The mobile node returns a binding update to the correspondent node.

The correspondent node then caches the mobile node's care-of address. This enables the optimal routing of subsequent packets from the correspondent node to the mobile node, which eliminates congestion at the mobile node's home agent and home link. It also reduces the impact of any possible failure of the home agent, the home link, or intervening networks leading to or from the home link, since these nodes and links are not involved in the delivery of most packets to the mobile node.

To operate as a correspondent node and to communicate with mobile nodes, enter the following TCP/IP management command:

```
$ TCPIP
TCPIP> sysconfig -r ipv6 mobileipv6_enabled=1
```

Use the `netstat` command with the `-s` option to display the contents of the mobile IPv6 binding cache.

1.2 NTP Version 4

This release of TCP/IP Services supports NTP Version 4 (NTP V4), incorporating new features and refinements to the NTP V3 algorithms. Except for symmetric mode in NTP Version 1, NTP Version 4 is backward compatible with older versions.

This section summarizes the differences between NTP V4 and NTP V3. For information about managing NTP, see Appendix B.

- Major code cleanup was completed for NTP Version 4.
- Most calculations are now done using 64-bit floating double format rather than 64-bit fixed-point format. The fixed-point format is still used with raw time stamps. The algorithms that process raw timestamps produce fixed-point differences before converting them to floating double format.
- The clock discipline algorithm has been redesigned to improve accuracy, reduce the impact of network jitter and allow an increase in poll intervals to well over one day. The NTP V4 design allows servers to increase the poll intervals even when synchronized directly to the peer. In NTP V3 the poll interval in such cases was fixed to the minimum (usually 64 seconds). For servers with hundreds of clients, the new design can dramatically reduce the network load.
- NTP V4 includes two new association modes that, in most applications, make per-host configuration unnecessary:
  - In multicast mode, a server sends a message at fixed intervals using specified multicast group addresses, while clients listen on these addresses. Upon receiving the message, a client exchanges several messages with the server in order to calibrate the multicast propagation delay between the client and server.
  - In manycast mode, a client sends a message to a specified multicast group address and expects one or more servers to reply. Using engineered algorithms, the client selects an appropriate subset of servers from the messages received and continues in ordinary client/server operation. Manycast mode provides better accuracy than multicast mode, without the price of additional network overhead.

  Both modes provide for automatic discovery and configuration of servers and clients without identifying servers or clients in advance.

- The following burst mode features are available:
  - Use the `iburst` keyword in the server configuration command when it is important to set the clock quickly when an association is first mobilized.
  - Use the `burst` keyword in the server configuration command when the network attachment requires an initial calling or training procedure.
- In all except a very few cases, all timing intervals are randomized, minimizing the tendency to self-synchronize and bunch messages, especially with a large number of configured associations.
- The arguments to the `enable` and `disable` commands are changed. Also, the `authenticate` command has been removed.
- A special control message is available to help reduce the level of spurious network traffic due to obsolete configuration files. If it is enabled, and a packet is denied service or exceeds the client limit, a compliant server sends the control message to the client. A compliant client will cease further transmission and send a message to the NTP log file.
- A filter algorithm reduces errors during asymetric delays (characteristic of PPP connections with telephone modems and downloading or uploading considerable traffic).
- The NTP V4 `ntpdc` utility does not work with previous versions of NTP. Previous versions of the `ntpdc` utility do not work with NTP V4.

1.3 BIND Version 9

The Domain Name System (DNS) maintains and distributes information about Internet hosts. DNS consists of a heirarchical databases containing the names of entities on the Internet, the rules for delegating authority over names, and mail routing information; and the system implementation that maps the names to Internet addresses.

In OpenVMS environments, DNS is implemented by the Berkeley Internet Name Domain (BIND) software. Compaq TCP/IP Services for OpenVMS implements a BIND server based on the Internet Software Consortium's (ISC) BIND Version 9 (BIND 9).

Note

BIND 9 is supported on Alpha systems only, and future support of BIND Version 8 (BIND 8) on VAX systems will be limited. Therefore, if you are using BIND 8 on a VAX system, Compaq recommends that you upgrade your BIND server to an Alpha system.

For information about managing BIND, refer to Appendix C.

1.3.1 BIND 9 Features

BIND 9 is a major rewrite of nearly all aspects of the underlying BIND architecture. Some of the important features of BIND 9 are:

- DNS security
  - DNSSEC (signed zones)
  - TSIG (signed DNS requests)

  - Access control lists
  - Dynamic update security policies
  - TKEY shared secrets
  - SIG(0) transaction signatures
- IPv6
  - Answers DNS queries on IPv6 sockets
  - IPv6 resource records (A6, DNAME, and so forth)
  - Bitstring labels
- DNS protocol enhancements
  - IXFR, DDNS, Notify, EDNS0
  - Improved standards conformance
- Views

  One server process can provide multiple views of the DNS name space (for example, an inside view to certain clients, and an outside view to others).
- Multiprocessor support
- Multithreading

Note

The BIND resolver is based on the BIND 8 implementation of DNS.

To take advantage of the multiprocessor and multithreading support provided with BIND 9, the OpenVMS SYSGEN parameter MULTITHREAD should be nonzero on multiprocessor systems. Note that this parameter is systemwide and affects other TCP/IP or OpenVMS components that use POSIX threads.

1.3.2 BIND 8 to BIND 9 Migration

BIND 9 is designed to be compatible with BIND 8. The following list summarizes the differences between them.

- Configuration file compatibility
  - BIND 9 supports most but not all of the TCPIP$BIND.CONF options of BIND 8.

    If your TCPIP$BIND.CONF file uses an unimplemented option, the BIND 9 server logs a warning message. A message is also logged about each option whose default has changed, unless the option is set explicitly in TCPIP$BIND.CONF.
  - The default of the `transfer-format` option has been changed from `one-answer` to `many-answers`. If you have slave servers running an old version of TCP/IP Services that does not understand the `many-answers` zone transfer format, you need to specify the following in either the `options` or `server` statement:

    ```
    transfer-format one-answer;
    ```

- In BIND 9, the BIND server will not start if it detects an error in TCPIP$BIND.CONF. Earlier versions of BIND would start despite errors, causing the server to run with a partial configuration. Errors detected during subsequent reloads do not cause the server to exit.

  Errors in master files do not cause the server to exit, but they do prevent the zone from being loaded.

- The set of logging categories in BIND 9 is different from that in BIND 8. If you have customized your logging on a per-category basis, you need to modify your `logging` statement to use the new categories.

  The `logging` statement takes effect only after the entire TCPIP$BIND.CONF file has been read. Therefore, when the server first starts up, any messages about errors in the configuration file are always logged to the TCPIP$BIND_RUN.LOG file, regardless of the contents of the `logging` statement. In BIND 8, the new logging configuration took effect immediately after the `logging` statement was read.

- The source address and port for Notify messages and Refresh queries is now controlled by the `notify-source` and `transfer-source` options, respectively, rather than by the BIND 8 `query-source` option.

- Multiple classes must be put into explicit views for each class.

- Zone file compatibility

  - BIND 9 complies strictly with the RFC 1035 and RFC 2308 rules regarding omitted time-to-live (TTL) values in zone files. Omitted TTL values are replaced by the value specified with the $TTL directive or, if there is no $TTL directive, by the previously specified TTL value.

    If there is no $TTL directive and the first resource record (RR) in the file does not have an explicit TTL field, the zone file is illegal because the TTL value of the first RR is undefined. BIND 4 and many versions of BIND 8 accept such files without warning and use the value of the SOA MINTTL field as a default for missing TTL values.

    BIND 9 emulates the nonstandard BIND 4/8 SOA MINTTL behavior and loads the file (provided the SOA is the first record in the file), but it also issues the following warning message:

    ```
    No TTL specified; using SOA MINTTL instead
    ```

    To avoid problems, use a $TTL directive in each zone file.

  - Some versions of BIND allow SOA serial numbers with an embedded period (for example, `3.002`), and converts the numbers into integers. This feature is not supported in BIND 9; serial numbers must be integers.

  - TXT records with unbalanced quotes (for example, `'host TXT "foo'`) do not cause errors in some versions of BIND. In BIND 9, if your zone files contain such records, potentially confusing error messages like the following are generated:

    ```
    Unexpected end of file
    ```

    This occurs because BIND 9 interprets everything up to the next quote character as a literal string.

- Some versions of BIND accept RRs that contain line breaks that are not properly quoted with parentheses, such as the following SOA:

```
@ IN SOA ns.example. hostmaster.example.
  ( 1 3600 1800 1814400 3600 )
```

  This is not legal master file syntax; BIND 9 treats it as an error. To correct the problem, move the opening parenthesis to the first line.

- The $$ construct for specifying a literal dollar sign ($) in a domain name is not recommended. Use the \$ construct instead.

- New protocol features
  - If you want to accept DNS queries over IPv6, you must specify the following in the TCPIP$BIND.CONF file:

```
listen-on-v6 {any; };
```

    This is not the default.

  - EDNS0

    BIND 9 uses EDNS0 to advertise its receive buffer size. It also sets an EDNS flag in queries to indicate it wants to receive DNSSEC responses.

    Most older servers that do not support EDNS0, including prior versions of BIND, send an error in response to these queries. When this happens, BIND 9 automatically retries the query without EDNS0.

    Certain non-BIND name server implementations may silently ignore these queries, instead of sending an error response. Name resolution is very slow, or fails, in zones where this type of server is used.

    When BIND 9 communicates with a server that supports EDNS0, such as another BIND 9 server, responses of up to 4096 bytes may be transmitted as a single UDP datagram, which is subject to fragmentation at the IP level. If a firewall incorrectly drops IP fragments, it can cause name resolution to slow down dramatically or fail.

  - Outgoing zone transfers now use the `many-answers` format by default. This format is not understood by old versions of BIND 4. Use the following option to correct this problem:

```
transfer-format one-answer;
```

    To prevent security problems, upgrade the slave servers.

  - Zone transfers to Windows 2000 DNS servers sometimes fail to properly handle DNS messages that are larger than 16K. To correct this problem, use the following option:

```
transfer-format one-answer;
```

- BIND 9 does not restrict the character set of domain names. It is fully 8-bit compatible.

  Host names published in the DNS should follow the rules set forth in RFC 952, but BIND 9 does not enforce the rules.

  Names containing unexpected characters cause security problems on systems that run certain applications that do not check data from the network sufficiently. Some earlier versions of BIND attempt to protect these applications from attack by discarding data containing characters deemed inappropriate in host names or mail addresses. This feature was controlled

by the check-names option in TCPIP$BIND.CONF. BIND 9 provides no such protection. Applications with these types of flaws should be upgraded.

- Server administration tools:
  - The ndc utility has been replaced by the rndc utility, which is capable of remote operation. Unlike ndc, rndc requires a configuration file. A template file is written to the directory pointed to by the TCPIP$ETC logical when you use the TCPIP$CONFIG.COM command procedure to enable the BIND server. The easiest way to generate a configuration file is to use the following command:

    ```
    $ rndc_confgen
    ```

  - The rndc utility does not work with BIND 8 name servers.
  - BIND 9 will not reload zones that allow dynamic updates.
  - The BIND 8 implementation of nsupdate separated update requests into multiple requests, based on the discovered zones that contained the records. In BIND 9, each update request must pertain to a single zone.

    To do multiple updates with a single invocation of nsupdate, terminate each update with an empty line or a send command.
- BIND 9 stores the authoritative data for each zone in a separate data structure. When a BIND 9 server is authoritative for both a child zone and its parent, it will have two distinct sets of NS records at the delegation point: the authoritative NS records at the child's apex, and a set of glue NS records in the parent.

  Unable to properly distinguish between these two sets of NS records, BIND 8 copied the child's NS records into the parent, causing the parent zone to be silently modified. Responses and zone transfers from the parent contained the child's NS records rather than the glue records configured into the parent (if any). For stub children, this behavior allowed the glue NS records to be omitted from the parent configuration.

  Sites that rely on this BIND 8 behavior need to add to the parent zone any omitted glue NS records and any necessary glue A records.

  Although stub zones can no longer be used as a mechanism for injecting NS records into their parent zones, they are still a useful way of directing queries for a given domain to a particular set of name servers.
- The DNSSEC and IPv6 features of BIND 9 are CPU intensive. Use large systems for these applications.
- BIND 9 is multithreaded, allowing full utilization of multiprocessor systems.
- The memory of the server has to be large enough to hold the cache and the zones. You can use the max-cache-size option to limit the amount of memory used by the cache, at the expense of reducing cache hit rates and causing more DNS traffic.

  Make sure than enough memory is available to load all zone and cache data into memory. To determine the best setting, wait until the name server has been in operation for a few weeks. The server process should reach a relatively stable size. Resource limits should be set higher than this stable size.
- Zone transfers no long run in a separate image (TCPIP$BIND_SERVER_XFER.EXE). They run in the context of a thread.

- For Alpha systems, the TCP/IP management command SET NAME /INITIALIZE has new behavior and execution requirements. Prior to TCP/IP Services Version 5.3, the command reloaded the BIND databases on the local host. Now the command reloads the BIND databases and the BIND configuration file.

  The SET NAME/INITIALIZE command now requires SYSPRV, BYPASS, or READALL privilege to be set on the user process. The command also now requires that either TCPIP$ETC:RNDC.CONF or TCPIP$ETC:RNDC.KEY be set up to allow secure communication between the user and the BIND server. To enable this functionality, enter the following sequence of commands:

  ```
  $ @SYS$MANAGER:TCPIP$DEFINE_COMMANDS.COM
  $ rndc_confgen -a
  $ @SYS$STARTUP:TCPIP$BIND_SHUTDOWN.COM
  $ @SYS$STARTUP:TCPIP$BIND_STARTUP.COM
  ```

  This procedure creates the TCPIP$ETC:RNDC.KEY file and restarts the BIND server so that it is aware of the newly created key file.

  Note

  These changes do not apply to the TCP/IP management command SET NAME/INITIALIZE on VAX systems.

- For Alpha systems, the TCP/IP management command SHOW NAME /STATISTICS has new behavior and execution requirements.

  Prior to TCP/IP Services Version 5.3, the command wrote statistics to the SYS$SPECIFIC:[TCPIP$BIND]TCPIP$BIND_SERVER_ STATISTICS.LOG file. The command now writes statistics information to the SYS$SPECIFIC:[TCPIP$BIND]TCPIP$BIND.STATS file.

  The SHOW NAME/STATISTICS command now requires SYSPRV, BYPASS, or READALL privilege to be set on the user process. The command also now requires that either TCPIP$ETC:RNDC.CONF or TCPIP$ETC:RNDC.KEY be set up to allow for secure communication between the user and the BIND server. To enable this functionality, enter the following sequence of commands:

  ```
  $ @SYS$MANAGER:TCPIP$DEFINE_COMMANDS.COM
  $ rndc_confgen -a
  $ @SYS$STARTUP:TCPIP$BIND_SHUTDOWN.COM
  $ @SYS$STARTUP:TCPIP$BIND_STARTUP.COM
  ```

  This procedure creates the TCPIP$ETC:RNDC.KEY file and restarts the BIND server so that it is aware of the newly created key file.

  Note

  These changes do not apply to the TCP/IP management command SHOW NAME/STATISTICS on VAX systems.

- The dynamic-update safety net mechanism and dynamic update related TCP/IP merge logicals no longer exist. Once a zone is configured to allow updates, the zone file should not be edited directly. To update a zone, use the `nsupdate` utility. If dynamic updates are enabled, zone files are written to disk periodically and purged automatically when the number of file versions exceeds five. Zones that are updated dynamically cannot be reloaded using the SET NAME/INITIALIZE command or the `rndc` utility.
- The round-robin scheduling of the BIND server has been changed. In previous versions of the BIND server, when multiple records were returned in an answer, they would get placed into the response in a round-robin manner for each consecutive request. With this version, a random round-robin ordering is used. The BIND server will randomly choose a starting point within the RRset and return the records in order starting at that point. There is currently no way to modify this behavior. The TCPIP$BIND_ROUND_ROBIN_OFF logical is ignored.

1.4 IMAP Server

The IMAP server for OpenVMS Mail and the Simple Mail Transfer Protocol (SMTP) server work together to provide reliable mail management in a client/server environment.

Note

IMAP is supported on Alpha systems only. Although images may appear on VAX systems after installation, these are not supported.

The IMAP server allows users to access their OpenVMS Mail mailboxes using client applications like Microsoft Outlook to view, move, copy, and delete messages. The SMTP server also allows the clients to create and send e-mail messages.

The IMAP server requires a certain level of the operating system. If you are running one of the following versions of OpenVMS, you must install the appropriate patch:

OpenVMS Version	Minimum Level Patch Kit
Alpha V7.2-1	VMS721_MAIL-V0100
Alpha V7.2-1H1	VMS21H1_MAIL-V0100
	VMS21H1_MAIL-V0200
Alpha V7.2-2	VMS722_MAIL-V0100
Alpha V7.3	VMS73_MAIL-V0100

OpenVMS versions higher than Version 7.3 automatically support the IMAP server without requiring any patches.

For more information about managing and using the IMAP server, refer to Appendix A.

1.5 Kerberos Enhancements to the TELNET Client and Server

Kerberos is freely available from the Massachusetts Institute of Technology (MIT), under a copyright permission notice. Kerberos for OpenVMS is supplied by Compaq Computer Corporation under the terms of the license from MIT. For more information about the Kerberos license, see the following web site:

`http://web.mit.edu/kerberos/www/`.

Kerberos is a network authentication protocol designed to provide strong authentication for client/server applications by using secret-key cryptography. Kerberos uses strong cryptography so that a client can prove its identity to a server (and vice versa) across an insecure network connection. The TCP/IP TELNET service uses Kerberos to make sure the identity of any user who requests access to a remote host is authentic.

Compaq TCP/IP Services for OpenVMS Version 5.3 supports the OpenVMS Kerberos Version 1.0 client, which is based on MIT Kerberos Version 5.

Before you can use the Kerberos TELNET client, the OpenVMS Security Client software must be configured on the OpenVMS system. For more information about installing and configuring the OpenVMS Security Client software, see the *Kerberos Version 1.0 for OpenVMS Security Client Installation Guide and Release Notes*.

The Kerberos Security Client kit contains copies of the MIT documentation listed in the *Kerberos Version 1.0 for OpenVMS Security Client Installation Guide and Release Notes*.

It is assumed that anyone using the Kerberos security features in TCP/IP has expert knowledge of Kerberos.

Note

Encryption is not supported in this version of TCP/IP Services.

1.5.1 Kerberos Principal Names

Before you use the Kerberos TELNET client, make sure the local host name is fully qualified in the local hosts database. Kerberos realms form principal names using fully-qualified domain names. For example, `terse.mbs.com` is a fully qualified domain name; `terse` is a simple host name.

Compaq TCP/IP Services for OpenVMS is usually configured so that the host name is entered in the hosts database as a simple host name. That is, on host TERSE, the TCP/IP management command SHOW HOST TERSE returns `terse`, not `terse.mbs.com`.

To correct a mismatch between the Kerberos realm and the TCP/IP Services configurations, follow these steps from a privileged account at a time when system usage is low:

1. Find the host's numeric address. For example:

```
$ TCPIP
TCPIP> SHOW HOST terse

     LOCAL database

Host address    Host name

15.28.311.11   terse
```

2. Remove the simple host name. For example:

```
TCPIP> SET NOHOST terse/CONFIRM
```

3. Use the SET HOST command to associate the fully qualified domain name with the IP address, as shown in the following example:

```
TCPIP> SET host "terse.mbs.com"/ADDRESS=15.28.311.11 -
_TCPIP> /ALIAS=("TERSE.MBS.COM", "terse", "TERSE")
```

   Specify the /ALIAS qualifier to ensure that applications can handle host names in uppercase and lowercase.

4. Confirm that the first name returned is fully qualified.

```
TCPIP> SHOW HOST terse

     LOCAL database

Host address    Host name

15.28.311.11   terse.mbs.com, TERSE.MBS.COM, terse, TERSE
```

1.5.2 Using the Kerberos TELNET Client

The following sections describe how to use the TELNET client to establish authenticated connections.

1.5.2.1 Initiating an Authenticated TELNET Connection

To initiate an authenticated connection, perform the following steps:

1. On a Kerberos-enabled system, enter a KINIT *username* command. Enter your password when prompted.

   Note

   Always specify the user name on the KINIT command line. Kerberos realms are usually set up with lowercase user names, but on OpenVMS, user names are stored in uppercase. When you specify the user name, it will be accepted as lowercase.

2. To initiate an authenticated connection, enter the following command:

```
$ TELNET/AUTHENTICATE host-name
```

3. To use the same ticket on a remote system, you can forward your ticket by entering the following command:

```
$ TELNET/AUTHENTICATE/FORWARD host-name
```

4. To use your credentials in another realm, enter the following command:

```
$ TELNET/AUTHENTICATE/REALM=realm-name.
```

1.5.2.2 TELNET Command Descriptions

This section describes the TELNET/AUTHENTICATE command.

TELNET/AUTHENTICATE

Qualifiers

/AUTHENTICATE
Optional. Default: None.

Specifies that you want the TELNET session to use Kerberos features.

Note

The /AUTHENTICATE qualifier also can be used with the TELNET commands OPEN and CONNECT.

/FORWARD
/NOFORWARD
Optional. Default: /NOFORWARD.

Forwards a copy of your Kerberos tickets to the remote host. The /NOFORWARD qualifier overrides any forwarding specified in your machine's configuration files. You must request forwardable tickets at the same time that you issue the KINIT command.

You must use the /AUTHENTICATE qualifier when you specify the /FORWARD qualifier.

/REALM=*realm-name*
Optional.

Requests Kerberos tickets for the remote host in the specified realm, instead of determining the realm itself.

You must use the /AUTHENTICATE qualifier when you specify the /REALM qualifier.

Examples

1.
```
$ TELNET/AUTHENTICATE/REALM=jet.mbs.com terse

%TELNET-I-TRYING, Trying ... 15.21.308.11
%TELNET-I-SESSION, Session 01, host terse, port 23
%TELNET-I-ESCAPE, Escape character is ^]
        terse.ucx.ttg.mbs.com
```

   This example logs in to system terse with Kerberos credentials.

2.
```
$ TELNET/AUTHENTICATE/FORWARD terse

%TELNET-I-TRYING, Trying ... 15.21.308.11
%TELNET-I-SESSION, Session 01, host terse, port 23
%TELNET-I-ESCAPE, Escape character is ^]
[Kerberos V5 accepts you as ''j_brown@terse.mbs.com'' ]
[Kerberos V5 accepted forwarded credentials ]
```

   This example forwards credentials to host terse for user j_brown.

1.5.3 Configuring the Kerberos TELNET Server

This version of TCP/IP Services supports a separate Kerberos TELNET server, in addition to the standard TCP/IP TELNET server.

The Kerberos TELNET server has the same major features as the TCP/IP Services TELNET server. However, there are minor differences between the two servers. For example, although the TELNET server supports IPv6 connections, the Kerberos TELNET server supports only the IPv4 protocol for communication with the Kerberos Key Distribution Center (KDC).

The TELNET server with Kerberos support is enabled by running the TCPIP$CONFIG.COM command procedure, as described in the *Compaq TCP/IP Services for OpenVMS Installation and Configuration* guide.

If the TELNET server is currently enabled and you want to support Kerberos, you must disable the TELNET service before you install this version of TCP/IP Services to ensure that the required TCPIP$TELNET user account and directory are created.

Note

Because the TELNET server will be stopped, do not use a TELNET connection to perform the following procedure.

To disable the Kerberos TELNET server, perform the following steps:

1. Invoke the TCPIP$CONFIG command procedure by entering the following command from a user account with system management privileges:

   ```
   $ @SYS$MANAGER:TCPIP$CONFIG.COM
   ```

2. On the Configuration menu, select the `Client components` option.
3. From the list of client components, select `TELNET`.
4. On the TELNET Configuration menu, select `Disable & Stop service on this node`.
5. Return to the Configuration menu.

For instructions on how to enable the Kerberos TELNET server, refer to the *Compaq TCP/IP Services for OpenVMS Installation and Configuration* manual.

1.5.3.1 Connecting to the Kerberos TELNET Server

The Kerberos TELNET server uses port 2323. Specify this port on the TELNET command line. For example:

```
$ TELNET/AUTHENTICATE terse.mbs.com /PORT=2323

%TELNET-I-TRYING, Trying ... 17.21.205.153
%TELNET-I-SESSION, Session 01, host terse.mbs.com, port 2323
-TELNET-I-ESCAPE, Escape character is ^]

 Welcome to OpenVMS (TM) Alpha Operating System, Version V7.3

Username:
```

1.6 Configuring Subsystem Attributes

TCP/IP Services supports UNIX subsystems and allows you to modify the attributes of those subsystems to change the way the TCP/IP Services software operates.

Subsystem configuration is provided for compatibility with Compaq *Tru64* UNIX. Compaq strongly advises you not to modify the attributes associated with subsystems except when the adjustment of an attribute is indicated (for example, to improve performance). In most cases, corresponding TCP/IP management commands are provided to help limit the side effects of modifying subsystem attributes.

The following sections describe how to display and modify the settings of the subsystem attributes. Modifying subsystem attributes without full knowledge of possible effects can cause unpredictable results and is recommended only as specifically directed by Compaq.

1.6.1 Displaying Subsystem Attributes and Values

You can use the `sysconfig -m` command to display static and dynamic subsystems, as follows:

```
$ TCPIP
TCPIP> sysconfig -m
cm: static
inet: static
iptunnel: static
ipv6: static
net: static
snmpinfo: static
socket: static
inetkvci: static
proxy: static
nfs: static
vfs: static
```

Depending on the configuration of your system, the list of subsystems displayed may differ from this example. There are two types of subsystems:

- **Static** subsystems are loaded at startup time and can be unloaded only when TCP/IP shuts down.
- **Dynamic** subsystems can be loaded and unloaded at will without shutting down and restarting TCP/IP Services.

Subsystems can be loaded but not available for use. To determine which subsystems are loaded, use the `sysconfig -s` command. This command displays the state of all subsystems. Subsystems can have the following states:

- Loaded and configured (available for use)
- Loaded and unconfigured (not available for use)

  This state applies only to static subsystems, which you can unconfigure, but you cannot unload.
- Unloaded (not available for use)

  This state applies only to loadable subsystems, which are automatically unloaded when you unconfigure them.

You can modify subsystem attributes at runtime, a change that will persist only as long as the system continues to run, or you can modify them in the system configuration table, preserving the changes through system reboots.

The persistence of a modified attribute value depends on the command or utility option you use, according to the following guidelines:

- For permanent modifications that persist across reboots, use the `sysconfigdb` utility, as described in Section 1.6.2.
- For temporary modifications that will not persist across reboots, use the `sysconfig -r` command as described in Section 1.6.3.

1.6.2 Modifying Subsystem Attributes in the System Configuration Table

To modify subsystem attributes so that changes persist across reboots, you must store the attribute's value in the system configuration table (TCPIP$ETC:SYSCONFIGTAB.DAT). This file is an ASCII text file, and is formatted in UNIX stanza file format. When a subsystem is loaded, the attributes that are not listed in the SYSCONFIGTAB.DAT file are set to their default values.

To modify subsystem attributes in the SYSCONFIGTAB.DAT file, follow these steps:

1. Create a stanza file as input to the procedure, as described in Section 1.6.2.1.
2. Use the `sysconfigdb` utility to update the system configuration table, as described in Section 1.6.2.2.
3. Reload the subsystem. A dynamic subsystem can be unloaded and reloaded using the `sysconfig` utility. A static subsystem is reloaded when the TCP/IP Services software is restarted.

Although you can edit the SYSCONFIGTAB.DAT file with any text editor, this practice is strongly discouraged. Syntax errors introduced when you edit the file can result in erroneous or unpredictable situations. Compaq recommends that you use the `sysconfig` utility to display the system configuration table and the `sysconfigdb` utility to modify its contents.

1.6.2.1 Creating a Stanza File

To add, update, or remove entries in the database, create a stanza file that contains the names and values for attributes that you want to modify.

The syntax for a stanza file entry is as follows:

```
entry-name:
       Attribute1-name = Attribute1-value
       Attribute2-name = Attribute2-value
       Attribute3-name = Attribute3-value1, Attribute3-value2
       .
       .
       .
```

The *entry-name* variable specifies the subsystem name.

The attributes for the subsystem are specified with the *Attribute1-name*, *Attribute2-name*, and *Attribute3-name* variables.

The values for the attributes are specified with the *Attribute1-value*, *Attribute2-value*, *Attribute3-value1*, and *Attribute3-value2* variables.

The stanza file syntax rules are as follows:

- Separate entries by one or more blank lines.
- A colon (:) terminates an entry name.
- A new line terminates an attribute name and value pair.
- Separate a attribute name and attribute value with an equals sign (=).
- Separate more than one attribute value with a comma (,).
- Entry names and attribute names can contain any printable character except spaces, new lines, and special characters, which must be specified appropriately.
- Entry attribute values can contain any printable character except new lines and special characters, which must be specified appropriately.
- Spaces and tabs are allowed at the beginning and at the end of lines.
- A pound sign (#) at the beginning of a line indicates a comment.
- Comments should be included only at the beginning or the end of an entry.

Several special quoting characters allow attribute values to contain special values and data representations. If you specify a quoting character, surround the attribute value with quotation marks. For example, to specify an octal value, use the backslash character:

```
\007
```

The TCPIP$ETC:SYSCONFIGTAB.DAT file is formatted as follows:

```
inet:
        inet_param1=inet_value1
        inet_param2=inet_value2

net:
        net_param1=net_value1
        net_param2=net_value2

proxy:
        proxy_param1=proxy_value1

socket:
        socket_param1=socket_value1
```

To modify a subsystem attribute, create a stanza file in your own directory. In the following example, the stanza file is named SOCKET_ATTRS.TXT.

```
$ TYPE SOCKET_ATTRS.TXT

socket:
         socket_param1 = socket_value1
$
```

1.6.2.2 Updating the System Configuration Table

After you create the stanza file, update the system configuration table using the sysconfigdb utility. To run the sysconfigdb utility, enter the following commands:

```
$ TCPIP
TCPIP> sysconfigdb
```

For information about using the sysconfigdb utility, refer to the command description in these release notes.

To update the system configuration table, use the sysconfigdb command with the -a option. Specify the stanza file on the command line using the -f option, as follows:

```
TCPIP> sysconfigdb -a -f stanza-filename subsystem
```

In this command line, *stanza-filename* is the file name of the stanza file that you created. The value for *subsystem* is the subsystem name for which you are changing an attribute.

The sysconfigdb command reads the specified file and updates the database. The modifications are made to the subsystem when it is reloaded.

For example, the following stanza file (TABLE_MGR.STANZA) defines the attributes for two subsystems, TABLE_MGR_1 and TBL_MGR_2.

```
$ TYPE TABLE_MGR.STANZA

table_mgr_1:
            size = 10
            name = Ten-Element-Table
tbl_mgr_2:
            size = 5
            name = Five-Element-Table

$
```

To add the contents of this stanza file to the system configuration table enter the following commands:

```
$ TCPIP
TCPIP> sysconfigdb -a -f table_mgr.stanza table_mgr_1

TCPIP> sysconfigdb -a -f table_mgr.stanza tbl_mgr_2
```

This example does not change the value of attributes on the running system. To modify the value of attributes in the running system, you must do one of the following:

- Use the sysconfig -u command to unload a dynamic subsystem, then use the sysconfig -r command to reload the subsystem.
- Stop and restart TCP/IP Services to reload static subsystems.

1.6.3 Modifying Subsystem Attributes at Run Time

You can modify a subsystem attribute using the sysconfig utility. This type of modification persists only during the current run session. If you shut down and reboot the system, the modification is lost.

For a description of the sysconfig utility, see the next section. For online HELP about sysconfig, enter the following commands:

```
$ TCPIP
TCPIP> HELP SYSCONFIG
```

To modify a subsystem attribute, enter the following command:

```
$ sysconfig -r attribute-name=attribute-value subsystem
```

The following sections describe the sysconfig and sysconfigdb utilities.

sysconfig

Maintains the subsystem configuration.

Format

sysconfig -c | -d | -m | -q | -Q | -r | -s | -u [*subsystem-name*] [*attribute-list*]

Description

The `sysconfig` command queries and modifies the in-memory subsystem configuration. Use this command to add subsystems, reconfigure subsystems that are already in memory, query subsystems, and unconfigure and remove subsystems.

The `sysconfig` utility allows you to modify the value of subsystem attributes, as long as the subsystem supports run-time modifications.

When you configure a subsystem using the `-c` flag, you make that subsystem available for use. If the subsystem is loadable, the `sysconfig` command loads the subsystem and then initializes the value of its attributes.

To modify the value of a subsystem attribute, use the `-r` (reconfigure) flag. Specify the subsystem attributes and values on the command line. The `sysconfig` utility modifies the named attributes by storing the value you specify in them. The modifications take effect immediately.

To get information about subsystem attributes, use either the `-q` flag or the `-Q` flag. You can specify an attribute list with both these flags. When you use the `-q` flag, the `sysconfig` command displays the value of attributes from the in-memory system configuration table. When you use the `-Q` flag, the `sysconfig` utility displays the following information about each attribute you specify in the attribute list or, if you omit the attribute list, every attribute for the specified subsystem.

- Attribute datatype.
- Operations supported by the attribute. For example, this information indicates whether you can reconfigure the attribute using the `sysconfig -r` command.
- Minimum and maximum allowed attribute values.

To get information about the state of subsystems, use the `-s` flag. This flag provides a list of the subsystems that are currently loaded and configured. If you specify *subsystem-name*, the command displays information about the state of that subsystem. Each subsystem can have one of three states:

- Loaded and configured (available for use)
- Loaded and unconfigured (not available for use but still loaded)

  This state applies only to static subsystems, which can be unconfigured but cannot be unloaded.
- Unloaded (not available for use)

  This state applies only to loadable subsystems, which are automatically unloaded when you unconfigure them with the `sysconfig -u` command.

Subsystems that are not being used can be unconfigured using the -u flag. Unconfiguring subsystems can free up kernel memory, making it available for other uses. You can unconfigure any static or loadable subsystem that supports run-time unconfiguration. If you unconfigure a loadable subsystem, that subsystem is also unloaded from the kernel.

You can use the sysconfig command to display the value of attributes on the local system. If you want to configure, reconfigure, or unconfigure a subsystem, you must be authorized to modify the kernel configuration. Only users who have a system group UIC or who have an account with SYSPRV, BYPASS, or OPER privilege can configure, reconfigure, or unconfigure the subsystems.

Parameters

subsystem-name
Specifies the subsystem on which you want to perform the operation. The *subsystem-name* argument is required for all flags except -s and -m. If you omit *subsystem-name* when you use the -s or -m flag, the sysconfig utility displays information about all loaded subsystems.

attribute-list
Specifies attribute names and, depending on the operation, attribute values.

- For reconfigure (-r) operations, the *attribute-list* argument has the following format:

  attribute1=value1 attribute2=value2...

  Do not include spaces between the attribute name, the equals sign (=), and the value.

- For query attribute (-q) operations, the *attribute-list* argument has the following format:

  attribute1 attribute2...

The *attribute-list* argument is required when you use the -r flag and is optional with the -q flag. Any attribute list specified with other flags is ignored by the sysconfig utility.

Flags

-c
Configures the specified subsystem by initializing its attribute values and, possibly, loading it into memory. Use this command whether you are configuring a newly installed subsystem or one that was removed using the sysconfig -u command option.

-d
Displays the attribute settings in the SYSCONFIGTAB.DAT file for the specified subsystem.

-m
Queries the mode for the specified subsystems. A subsystem's mode can be static or dynamic. If you omit the subsystem name, sysconfig displays the mode of all the configured subsystems.

-q
Queries attribute values for the configured subsystem specified by *subsystem-name*. If you omit the attribute list, values for all the specified subsystem's attributes are displayed.

-Q
Queries information about attributes of the configured subsystem specified by subsystem-name. The information includes the attribute data type, the operations supported, and the minimum and maximum values allowed for the attribute. Note that the minimum and maximum values refer to length and size for attributes of char and binary types, respectively. If you omit the attribute-list argument, information about all attributes in the specified subsystem is displayed.

-r
Reconfigures the specified subsystem. You must supply the subsystem name and the attribute list when you use this flag.

-s
Queries the subsystem state for the specified subsystems. If you omit the subsystem name, sysconfig displays the state of all the configured subsystems.

-u
Unconfigures and, if the subsystem is loadable, unloads the specified subsystem from the kernel.

Examples

The following examples show how to use the sysconfig command.

1.
```
TCPIP> sysconfig -s
inet: loaded and configured
net: loaded and configured
socket: loaded and configured
iptunnel: loaded and configured
ipv6: loaded and configured
snmpinfo: loaded and configured
```

This example shows how to display the subsystems and their status.

2.
```
TCPIP> sysconfig -q net
net:
ifnet_debug = 0
ifqmaxlen = 1024
lo_devs = 1
lo_def_ip_mtu = 4096
nslip = 0
```

This example shows how to display subsystem attributes and their values.

3.
```
TCPIP> sysconfig -s net
net: loaded and configured
```

This example shows how to query the state of a particular subsystem.

sysconfigdb

Manages the subsystem configuration database.

Format

sysconfigdb {-a | -u} [-t *target*] -f *file subsystem-name*

sysconfigdb {-m | -r} [-t *target*] -f *file* [subsystem-name]

sysconfigdb -d [-t *target*] *subsystem-name*

sysconfigdb -l [-t *target*] [*subsystem-name*,...]

Description

The `sysconfigdb` utility is used to manage the subsystem configuration table (TCPIP$ETC:SYSCONFIGTAB.DAT). However, it can also be used to maintain any text file that has the same format as the SYSCONFIGTAB.DAT file. The file being managed by the `sysconfigdb` utility is called the target file. By default, the target file is the SYSCONFIGTAB.DAT file. To specify another file as a target file, use the `-t` flag.

To modify a target file, create a stanza file. This stanza file contains the name of one or more subsystems, each with a list of attributes and their values, as described in Section 1.6.2.1.

When the target file is the SYSCONFIGTAB.DAT file, modifications you make to it are synchronized into the subsystem configuration table, but the subsystems are unchanged until the next time they are loaded.

When the target file is another file, there is no synchronization with the subsystem configuration database.

Restrictions

You must have system management privileges to run the `sysconfigdb` utility to modify the system configuration table.

Parameters

subsystem-name
Specifies a subsystem that contains the attributes you want to modify. The subsystem name and attributes are in a stanza input file.

You must specify the subsystem name when deleting (`-d`), adding (`-a`), or replacing (`-u`) a subsystem.

In other cases, when you do not specify a subsystem name, the operation is attempted for all the subsystems and attributes specified in the input file.

Flags

-a
Adds the specified subsystem entry to the target file.

-d
Deletes the specified subsystem entry from the target file.

-f *file*
Specifies the input file, a stanza file that contains entries for one or more subsystems. The default target file is the SYSCONFIGTAB.DAT file. Specify another target file by using the -t target flag.

-l
Lists the specified subsystem entries in the target file. If you do not specify a subsystem name, all subsystem entries in the target file are listed. The SYSCONFIGTAB.DAT file is the default target file.

-m
Merges subsystem attributes specified in the input file with the subsystem attributes in the target file. If you do not specify a subsystem name, all subsystem entries in the input file are merged. The SYSCONFIGTAB.DAT file is the default target file.

-r
Removes the subsystem entries specified in the input file from the target file. The only entries removed are those that have attribute names and values that exactly match those in the input file. If you do not specify the subsystem name, all subsystem entries in the input file with attributes that match are removed from the target file. The SYSCONFIGTAB.DAT file is the default target database file.

-t *file*
Specifies the target file for the operation. If you do not specify this flag, the default target file is the SYSCONFIGTAB.DAT file.

-u
Replaces a subsystem entry in the target file with the subsystem entry specified in the input file.

Examples

The following examples show how use the sysconfigdb utility.

1.
```
$ TCPIP
TCPIP> sysconfigdb -u -f table_mgr.stanza table_mgr_1
```

   This command replaces the table_mgr_1 entry in the SYSCONFIGTAB.DAT file with the information in the TABLE_MGR.STANZA file for the table_mgr_1 subsystem. The command updates the in-memory copy of the subsystem configuration database to match the modified SYSCONFIGTAB.DAT file.

2.
```
TCPIP> sysconfigdb -m -f table_mgr.stanza tbl_mgr_2
```

   This command merges the tbl_mgr_2 information from the table_mgr.stanza file with the information already in the tbl_mgr_2 entry in the SYSCONFIGTAB.DAT file. The command updates the in-memory copy of the subsystem configuration database to match the modified SYSCONFIGTAB.DAT file.

3.
```
TCPIP> sysconfigdb -l table_mgr_1
table_mgr_1:
             size = 10
             name = Ten-Element-Table
```

   This command lists the entry for the subsystem table_mgr_1. This command does not update the in-memory copy of the subsystem configuration database.

4.
```
TCPIP> sysconfigdb -d table_mgr_1
```

   This command deletes the table_mgr_1 entry from the SYSCONFIGTAB.DAT file and updates the in-memory copy of the subsystem configuration database to match the modified SYSCONFIGTAB.DAT file.

1.7 Online Help for Error Messages

This release of TCP/IP Services provides additional online Help for error messages. You can now access Help for messages issued during product and service operations, such as component startup and shutdown.

For information about setting up and using the TCP/IP Services HELP message database, see Section 2.8.

1.8 LPD Server Cluster Support

This release of TCP/IP Services features enhancements to the LPD server to improve network printing in an OpenVMS Cluster environment. This section describes the appropriate changes to management procedures.

1.8.1 Implementing Clusterwide Print Queues

To implement clusterwide print queues, set up the following generic print queues and execution queues:

- Incoming print queues

  The TCPIP$LPD_QUEUE execution queue is replaced by the new TCPIP$LPD_IN generic queue and one or more execution queues for each node in the cluster. You can specify the number of execution queues per node using the `Inbound-Queues-Per-Node` configuration option, as described in Section 1.8.4. By default, on execution queue is automatically created for each node in the cluster.

  When the LPD server starts, the appropriate number of execution queues are automatically created and named TCPIP$LPD_IN_*nodename_nn*, where *nodename* is the cluster node's SCS name, and *nn* is the number of the execution queue within the set of execution queues on that node. The TCPIP$LPD_IN generic queue refers to the execution queues by number (that is, all the first execution queues on all the nodes, followed the second, and so forth), thus achieving load balancing across all the nodes in the cluster.

- Utility print queues

  LPD utility queues are outbound execution queues for printers on remote LPD hosts. The generic queue TCPIP$LPD_OUT can point to one or more outbound execution queues for each node in the OpenVMS Cluster, named TCPIP$LPD_OUT_*nodename_nn*, where *nodename* is the SCS node name of the cluster node, and *nn* is the number of the queue on that node.

  By default, outbound execution queues are not created automatically when TCP/IP Services starts up. You must specify the creation of outbound execution queues, using the `Utility-Queues-Per-Node` configuration option, as described in Table 1–2.

  As with the inbound execution queues, the TCPIP$LPD_OUT generic queue points to the execution queues by number, thus achieving load balancing.

  The `printcap` attributes of the utility queues are defined by default as follows:

```
TCPIP$LPD_OUT_nodename_nn:\
:lf=/TCPIP$LPD_ROOT/000000/TCPIP$LPD_OUt_nodename_nn.LOG:\
:lp=TCPIP$LPD_OUT_nodename_nn:\
:rm=localhost:\
:sd=/TCPIP$LPD_ROOT/TCPIP$LPD_OUT_nodename_nn:\
```

  Entries in the `printcap` file are required only if you want to change one of these default settings.

1.8.2 Using Clusterwide Print Queues

Print jobs are queued to the TCPIP$LPD_OUT print queue. To specify the printer on the PRINT command line, include the following qualifiers.

- /PARAMETER=(HOST=*hostname*), where *hostname* is the name of the remote LPD host
- /PARAMETER=(PRINTER=*printername*), where *printername* is the name of the printer on the remote LPD host.

For example, to print your LOGIN.COM file on the printer named XYZPRINT on the host LPDSVR.XYZ.ORG, enter the following command:

```
$ PRINT/QUEUE=TCPIP$LPD_OUT/PARAMETER=(HOST=LPDSVR.XYZ.ORG,PRINTER=XYZPRINT)-
_$ SYS$LOGIN:LOGIN.COM
```

You might want to associate DCL symbols with the destination printers, creating command names that are easy to remember. The new command names can be made available systemwide by including them in the system SYLOGIN.COM file.

The printer specified in the preceding example can be defined with the following command:

```
$ XYZPRINT :== $ PRINT/QUEUE=TCPIP$LPD_OUT-
_$ /PARAMETER=(HOST=LPDSVR.XYZ.ORG,PRINTER=XYZPRINT)
```

If the logical name is defined systemwide, the XYZPRINT command always prints to the specified printer on the specified host.

1.8.3 Defining the LPD Spooler Directory

The TCPIP$LPD_SPOOL logical name is replaced by the TCPIP$LPD_ROOT logical name. The new logical name defines the LPD root directory; if not specified, the logical name points by default to the same directory as the old TCPIP$LPD_SPOOL logical name, SYS$SPECIFIC:[TCPIP$LPD]. You can redefine the LPD root directory by defining TCPIP$LPD_ROOT, as follows:

```
$ DEFINE/SYSTEM/EXECUTIVE_MODE/TRANSLATION_ATTRIBUTES=-
_$ (CONCEALED,TERMINAL) TCPIP$LPD_ROOT dev:[directory.]
```

The printcap file need not be changed when you define the LPD root directory. The root directory is defined in the printcap file using the following entry:

```
:sd= /TCPIP$LPD_ROOT/000000/MYQUEUE:\
```

Inbound execution queues do not have printcap entries; rather, they take on the characteristics of the local queues to which they submit print jobs.

1.8.4 Configuring the LPD Server

The logical names used to modify LPD configuration information are generally replaced by entries in the TCPIP$LPD.CONF file, a text file that you can modify with any text editor.

Table 1–2 describes the TCPIP$LPD.CONF options.

Table 1–2 LPD Configuration Options and Descriptions

Configuration Option	Description
1st-VFC-Prefix-Special	Specifies not to insert an extra line feed character at the beginning of print files.
Droptime	Indicates how long after repeated timeouts a connection should be maintained before closing the connection. The value is specified in seconds. The Drop timer is in effect only after the link has been established, and it takes effect only if the Keepalive configuration option is set. The default value for the Drop timer is 300 seconds.
Idle-Timeout	Specifies the length of time for the LPD server to wait for an incoming LPD connection, in OpenVMS delta time format. The default is 5 minutes. This behavior requires that the Persistent-Server option be specified.
Inbound-Queues-Per-Node	Specifies the number of inbound execution queues to create for each cluster node when the LPD server starts. The default is 1.
Keepalive	Specifies the number of seconds to wait before checking the other end of a link that appears to be idle. The Keepalive timer detects when a remote host has failed or has been brought down, or when the logical connection has been broken.
Loop-Max	Specifies the maximum number of times the LPD server should retry a connection. The default is no maximum (the same as setting this option to 0). This behavior requires that the Persistent-Server option be specified.
Persistent-Server	Enables the persistence of the LPD server. This behavior is disabled by default.
Probetime	Specifies the number of seconds to wait before timing out the connection. The value of the Probetime option must always be less than or equal to the value of the Droptime option. The default value for the Probetime option is 75 seconds. The Probe timer controls: • When establishing an initial connection, the number of seconds TCP/IP Services will wait for a response before a timeout occurs. The time is active regardless of whether the Keepalive configuration option is set. • The length of time (in seconds) allowed to pass before TCP/IP Services checks an idle connection. This requires that the Keepalive configuration option be set.
PS-Extensions	Controls Compaq PrintServer extension support. By default, PrintServer extensions are supported by LPD. To disable support, specify the NON_PS keyword to this option. To enable support, specify the LPS keyword.
Retry-Interval	Specifies the amount of time to wait before requeuing a print job that failed because of a soft error, such as the loss of the TCP connection. The default is 5 minutes (0 00:05:00.00).

(continued on next page)

Table 1–2 (Cont.) LPD Configuration Options and Descriptions

Configuration Option	Description
Retry-Maximum	Specifies the OpenVMS delta time for which the LPD symbiont will continue to requeue a print job that has failed with a soft error. The default is 1 hour (0 01:00:00.00).
Setup-NoLF	By default, the LPD server inserts a line feed into the byte stream after the SETUP module and before the actual print file. This option allows you to control this behavior. To prevent LPD from inserting line feed characters, set this option to TRUE. For information about controlling this behavior using the printcap file, see Section 1.8.5.
Stream-Passall	Controls whether LPD will add extra line feed characters to files with embedded carriage control (the default). Set this option to preserve the behavior of previous versions of TCP/IP Services. This is useful when your users print from Compaq *PATHWORKS* Client software.
Utility-Queues-Per-Node	Specifies the number of outbound execution queues to create for each cluster node when the LPD server starts. The default is 0.
Synchronize-All-Jobs	Controls whether the the LPD print symbiont process running in an inbound execution queue (TCPIP$LPD_IN_*nodename_nn*) will synchronize on the completion of each job that it submits to a final destination print queue. If the LPD service log option LOGOUT is set using the TCP/IP management command SET SERVICE/LOG, when a print job submitted by the symbiont process completes, the LPD server synchronizes and sends an OPCOM message containing the job number, queue name, and user and host names of the submitter. Each synchronization causes the consumption of one slot of the symbiont process's AST quota and some dynamic memory. In situations where many jobs submitted by an LPD symbiont process are pending (for example, because the print queue to which they were submitted has been stopped) the symbiont process can exhaust its AST quota or virtual memory. If the Synchronize-All-Jobs option is set to FALSE, synchronization occurs only for print jobs that have either an LPD mailback completion notice or a temporary layup file sent from the LPD client to be used in the printing of the job. Setting this option to FALSE helps limit the exhaustion of dynamic memory or AST quota when many print jobs are outstanding, because most print jobs do not use mailback completion (/PARAMETERS=MAIL) or layup files (/PARAMETERS=LAYUP_DEFINITION). The default setting for the Synchronize-All-Jobs option is TRUE, which is appropriate for most sites. Systems with heavy inbound processing across many print queues might need to set this option to FALSE.
VMS-Flagpages	Enables the OpenVMS flag-page print options described in the *Compaq TCP/IP Services for OpenVMS Management* guide.

1.8.5 Using the printcap File to Prevent Line Feed Insertion

To prevent the LPD server from inserting a line feed into the byte stream after the SETUP module and before the actual print file, include the `sn` symbol in the `printcap` printer configuration file using the TCPIP$LPRSETUP program as described in the *Compaq TCP/IP Services for OpenVMS Management* guide.

Including this `sn` symbol prevents LPD from inserting the line feed character on a per-queue basis, overriding the definition of the `Setup-NoLF` configuration option in the TCPIP$LPD.CONF file (described in Table 1–2).

1.8.6 Configuring a High-Availability LPD Server

You can use the new LPD server cluster features to provide a high-availability, load-balanced LPD server. To configure this, create a cluster alias for all of the IP interfaces of your LPD server nodes. On your LPD clients, specify the cluster alias as the LPD server to which to send LPD jobs. For more information about load balancing and the load broker, refer to the *Compaq TCP/IP Services for OpenVMS Management* guide,

1.8.7 Migrating to the Clusterwide LPD Server

The LPRSETUP utility uses clusterwide configuration definitions in new `printcap` entries that it creates for you, but it does not automatically change existing entries that refer to the node-specific directory. This section describes how to migrate an existing LPD configuration to the cluster.

If you have been using LPD and want to migrate your current environment to work in a cluster, you must perform some manual conversion steps. If you do not, your existing LPD configuration will continue to work as it always has, in single-node mode.

To migrate your system-specific LPD configuration to a clusterwide configuration:

1. Shut down LPD by entering the following command:

   ```
   $ @SYS$STARTUP:TCPIP$LPD_SHUTDOWN.COM
   ```

2. Create a clusterwide directory to serve as your LPD root directory. For example:

   ```
   $ CREATE/DIRECTORY dev:[directory]/OWNER=TCPIP$LPD
   ```

   For the disk device (*dev:*), specify a clusterwide device. For *directory*, specify a directory on the clusterwide device (for example, [LPD_ROOT]).

3. Add a definition of the TCPIP$LPD_ROOT logical to your system startup command file, pointing the logical to your new LPD root directory.

   The logical must either be defined before TCPIP$STARTUP.COM runs or in TCPIP$LPD_SYSTARTUP.COM. If you define TCPIP$LPD_ROOT in TCPIP$LPD_SYSTARTUP.COM, make sure the logical is defined before the DCL command that starts your client queues. If you define TCPIP$LPD_ROOT in TCPIP$LPD_SYSTARTUP, the following informational message is displayed when you start LPD:

   ```
   %DCL-I-SUPERSEDE, previous value of TCPIP$LPD_ROOT has been superseded
   ```

   This message reflects that fact that TCPIP$LPD_ROOT was defined before TCPIP$LPD_SYSTARTUP.COM was invoked. Therefore, your definition of TCPIP$LPD_ROOT in TCPIP$LPD_SYSTARTUP.COM will supersede the one made by TCPIP$LPD_STARTUP.COM. If you are defining TCPIP$LPD_ROOT in TCPIP$LPD_SYSTARTUP.COM and you

do not see this informational message, there may be an error in your definition of TCPIP$LPD_ROOT. Make sure you included the /SYSTEM and /EXECUTIVE_MODE qualifiers.

Your definition of the TCPIP$LPD_ROOT logical should reside in a clusterwide file. Create a clusterwide command file to be invoked from TCPIP$LPD_SYSTARTUP.COM. For example, create a file named LPD_ROOT.COM in the directory that is used to store clusterwide system management files (in this case, COMMON_MANAGER), containing the command that defines TCPIP$LPD_ROOT.

```
$ TYPE COMMON_MANAGER:LPD_ROOT.COM
$ DEFINE/SYSTEM/EXECUTIVE_MODE -
_$ /TRANSLATION_ATTRIBUTES=(CONCEALED,TERMINAL) TCPIP$LPD_ROOT DISK1:[LPD_ROOT.]
```

Include the command to invoke the new command file at the beginning the TCPIP$LPD_SYSTARTUP.COM file. For example:

```
$ @COMMON_MANAGER:LPD_ROOT.COM
```

4. Define the TCPIP$LPD_ROOT logical to point to the directory you just created. For example:

```
$ DEFINE/SYSTEM/EXECUTIVE_MODE -
_$ /TRANSLATION_ATTRIBUTES=(CONCEALED,TERMINAL) TCPIP$LPD_ROOT DISK1:[LPD_ROOT.]
```

5. Copy your current printcap file to your new LPD root directory. For example:

```
$ COPY SYS$SPECIFIC:[TCPIP$LPD]TCPIP$PRINTCAP.DAT TCPIP$LPD_ROOT:[000000]
```

6. Edit the new printcap file and change all occurrences of /SYS$SPECIFIC/TCPIP$LPD to TCPIP$LPD_ROOT/000000.

7. Create subdirectories for each of the printcap entries defined in your printcap file.

   Create a subdirectory for each subdirectory under your old LPD home directory. Use the following command to list the subdirectories:

   ```
   $ DIRECTORY SYS$SPECIFIC:[TCPIP$LPD]*.DIR
   ```

   For example, if you had an existing LPD client queue called LASERJET with a spooler directory of SYS$SPECIFIC:[TCPIP$LPD.LASERJET], enter the following command to create the new clusterwide subdirectory:

   ```
   $ CREATE/DIRECTORY DISK1:[LPD_ROOT.LASERJET]
   ```

8. Enable LPD on each cluster node. First, define TCPIP$LPD_ROOT as you have it defined in your system startup sequence. Then run the TCPIP$CONFIG.COM command procedure on each of the cluster nodes on which you want to run LPD.

   From Server menu, choose the option to enable LPD.

9. If you have used the LPD configuration logical names and you want to copy the settings to the new configuration, or if you want to change the default values for the new configuration options, modify the LPD configuration file (TCPIP$LPD.CONF), as described in Section 1.8.4.

   The LPD configuration file is optional; defaults are provided for each configurable item. For more information about copying logical name settings to configuration options, see Section 1.8.7.1.

10. If you are using communication proxies to authorize LPD print users, and the TCPIP$PROXY database is not shared by all members of the OpenVMS Cluster, you must add the proxy entries on each node that is running the LPD server.

    To determine whether communication proxies are used, enter the following TCP/IP management command:

    ```
    TCPIP> SHOW SERVICE LPD/FULL
    ```

    If application proxies have been enabled, the Aprox flag appears in the list of flags.

    To determine whether the proxy database is shared by the cluster nodes, enter the following command:

    ```
    $ SHOW LOGICAL TCPIP$PROXY
    ```

    This logical name points to the location of the proxy database. If it is not on a clusterwide device, and the Aprox flag is set for the LPD service, then you must add proxy entries on each node running the LPD service.

    To display the proxy entries, enter the TCP/IP management command SHOW PROXY. Then use the SET PROXY command to enter the application proxy information on each node.

    To load proxy information on all cluster members, restart TCP/IP Services.

11. Start LPD on each cluster node by entering the following command:

    ```
    $ @SYS$STARTUP:TCPIP$LPD_STARTUP.COM
    ```

1.8.7.1 Migrating Configuration Options

Table 1–3 lists the LPD logical names and the associated options in the TCPIP$LPD.CONF file.

Table 1–3 Logical Names and LPD Configuration Options

Logical Name	TCPIP$LPD.CONF Option Name
TCPIP$LPD_PERSISTENT_SERVER	Persistent-Server
TCPIP$LPD_IDLE_TIMEOUT	Idle-Timeout
TCPIP$LPD_LOOP_MAX	Loop-Max
TCPIP$LPD_KEEPALIVE	Keepalive
TCPIP$LPD_PROBETIME	Probetime
TCPIP$LPD_DROPTIME	Droptime
TCPIP$LPD_SETUP_NOLF	Setup-NoLF
TCPIP$LPD_1ST_VFC_PREFIX_SPECIAL	1st-VFC-Prefix-Special
TCPIP$LPD_VMS_FLAGPAGES	VMS-Flagpages
TCPIP$LPD_PS_EXT	PS-Extensions
TCPIP$LPD_STREAM_PASSALL	Stream-Passall
TCPIP$LPD_RETRY_INTERVAL	Retry-Interval
TCPIP$LPD_MAXIMUM_INTERVAL	Retry-Maximum

(continued on next page)

Table 1–3 (Cont.) Logical Names and LPD Configuration Options

Logical Name	TCPIP$LPD.CONF Option Name
TCPIP$LPD_*qname*_SETUP_NOLF	None. This characteristic is defined using the new `sn` symbol in the `printcap` file. For more information, see Section 1.8.5.

Table 1–4 describes the logical names that continue to be valid:

Table 1–4 Valid LPD Logical Names

Logical Name	Description
TCPIP$LPD_ROOT	Replaces TCPIP$LPD_SPOOL.
TCPIP$LPD_SYMB_DEBUG	Replaces TCPIP$LPD_DEBUG and LPD_DEBUG.
TCPIP$LPD_RECV_DEBUG	Replaces TCPIP$LPD_RCV and LPD_RCV.

Table 1–5 lists the logical names that are obsolete.

Table 1–5 Obsolete LPD Logical Names

Logical Name	Replacement
TCPIP$LPD_PRINTCAP	No replacement. The `printcap` file is named TCPIP$PRINTCAP.DAT and is stored in the LPD root directory.

TCPIP$LPD_LOGFILE	No replacement. The log files are named for the execution queues and are stored in the LPD root directory in these formats:
	`TCPIP$LPD_ROOT:TCPIP$LPD_IN_`*`nodename_nn`*`.LOG`
	`TCPIP$LPD_ROOT:TCPIP$LPD_OUT_`*`nodename_nn`*`.LOG`

TCPIP$LPD_SPOOL	Replaced by TCPIP$LPD_ROOT.
TCPIP$LPD_DEBUG LPD_DEBUG	Replaced by TCPIP$LPD_SYMB_DEBUG.
TCPIP$LPD_RCV LPD_RCV	Replaced by TCPIP$LPD_RECV_DEBUG.
TCPIP$LPD_CLIENT_ENABLE	No replacement.

1.9 UNIX Services Database File

This version of TCP/IP Services provides an editable text file, TCPIP$ETC:SERVICES.DAT, that allows you to associate Internet service names and aliases with the port number and protocol, in the same way that they are associated on UNIX systems using the `/etc/services` file. This provides network programmers with the ability to refer to services by name rather than hard-coded port numbers.

When you configure the BIND resolver using the TCPIP$CONFIG command procedure, the SERVICES.DAT file is written to the directory pointed to by the TCPIP$ETC logical name. This template file contains information about the format of the file.

The following Compaq C routines are defined to operate on the SERVICES.DAT file:

- `getservbyname()`
- `getservbyport()`
- `getservent()`
- `setservent()`
- `endservent()`

The `getservbyname()` and `getservbyport()` functions look first in the traditional (RMS indexed) services database that is referenced by the TCPIP$SERVICE logical name. If the requested service is not found there, TCPIP$ETC:SERVICES.DAT is searched before returning an error to the caller.

Note

The TCPIP$ETC:SERVICES.DAT file is not compatible with TCP/IP management commands. Service definitions established with the SET SERVICE command are not stored in the TCPIP$ETC:SERVICES.DAT file. Services listed in the TCPIP$ETC:SERVICES.DAT file are not shown by the SHOW SERVICE command.

1.10 NFS Support for Extended File Specifications

The NFS server and the NFS client support OpenVMS extended file specifications (EFS) on ODS-5 disk volumes.

You can use NFS server to export files on OpenVMS ODS-5 volumes. The traditional ODS-2 volumes continue to be supported. The NFS client can emulate an ODS-5 volume.

Note that the NFS server and NFS client support the ISO Latin-1 character set only.

If an ODS-5 volume is mapped and exported, the NFS server automatically supports EFS features and ignores the NAME_CONVERSION option if it is specified in the export record.

On ODS-2 volumes (with or without the NAME_CONVERSION option), files with all uppercase names are displayed on non-OpenVMS clients with all lowercase letters. On ODS-5 volumes, the file names are displayed by clients in the same case as they are displayed locally on the server host.

If an ODS-2 volume contains file names that were created using the NFS NAME_CONVERSION option and include lowercase or special characters that are invalid for ODS-2 file names, those file names displayed locally on the server host contain escape codes, as described in the *Compaq TCP/IP Services for OpenVMS Management* manual. If a SET VOLUME /STRUCTURE_LEVEL=5 command is performed on this volume, the names are displayed by clients with the escape codes exactly as they are displayed locally on the server host.

1.10.1 Enabling Extended File Specifications

Extended file specifications are provided by the ODS-5 file system. To mount an ODS-5 volume, add the /STRUCTURE=5 qualifier to TCP/IP management command MOUNT. For example:

```
$ TCPIP
TCPIP> MOUNT DNFS0: BOOK1 BEATRICE -
_TCPIP> /PATH="/INFERNO" /HOST="FOO.BAR.EREWHON" -
_TCPIP> /STRUCTURE=5 /SYSTEM
```

The /STRUCTURE qualifier accepts the following values:

- 5 to indicate ODS-5
- 2 to indicate ODS-2 (the default)

For more information about the MOUNT/STRUCTURE command, display the online Help by entering the following command:

```
TCPIP> HELP MOUNT/STRUCTURE
```

Note

When you display device information using the DCL command SHOW DEVICE/FULL, the NFS disk is incorrectly shown as being accessed by DFS. For example:

```
$ SHOW DEVICE/FULL
 . . .
Disk DNFS1:, device type Foreign disk type 7, is online, mounted,
file-oriented device, shareable, accessed via DFS
 . . .
```

1.10.2 NFS Client Support for Extended File Specifications

If you do not include the /STRUCTURE qualifier on the MOUNT command, the NFS client assumes that the file system structure being accessed is an ODS-2 volume. You can change this default by defining the following logical name:

```
TCPIP$NFS_CLIENT_MOUNT_DEFAULT_STRUCTURE_LEVEL
```

You can use this logical name to ensure that all NFS disks on the system have ODS-5 support enabled. Set the value of the logical to 2 for ODS-2 (the default), or 5 for ODS-5. To override this logical, include the /STRUCTURE qualifier to the TCP/IP management command MOUNT.

The NFS client supports the extended character set supported by the OpenVMS operating system. The NFS client does not support NUL (ASCII 0). The length of a file name is limited to 232 characters, including the file name, dot, file extension, semicolon, and version number.

Refer to the OpenVMS product documentation for more information about extended file specification support.

1.11 FTP Server and FTP Client Support for UNIX Path Names (Alpha Only)

The FTP server and the FTP client have been enhanced to support UNIX path names. The FTP client can be used to access files using UNIX paths, and the FTP server can interpret the path names.

1.11.1 Specifying UNIX Path Names with FTP

For ODS-5 volumes, the FTP client and FTP server accept a path name argument in UNIX format. The following FTP commands accept UNIX path names:

- APPEND
- DELETE
- DIRECTORY
- GET
- PUT
- RENAME
- VIEW

For display, most UNIX path names are translated to OpenVMS format (for example, with the DIRECTORY command). However, UNIX path names are not translated to OpenVMS format in the following type of message:

`150 Opening data connection for` *`file-name IP-address`*

For the FTP client and server to support these ODS-5 features, the Compaq C shareable library (SYS$SHARE:DECC$SHR.EXE) must be updated to an ECO built after October 2000, including:

- VMS712_ACRTL-V0200
- VMS721H1_ACRTL-V0200
- VMS721_ACRTL-V0300
- VMS73_ACRTL-V0100

1.11.2 Specifying Special Characters

The following list describes the way certain characters are handled on ODS-5 volumes.

- All characters supported by ODS-5 volumes are valid. When you specify files in UNIX format, do not use the caret (^) escape character to quote special characters. For example, specify `funny[path name.txt.bkp;1`, not `funny^[path name^.txt.bkp;1`.
- The question mark (?) and asterisk (*) characters are interpreted as UNIX wildcard characters. OpenVMS interprets these characters as single-character wildcard characters. The OpenVMS command interpreter accepts the percent sign (%) in the same way. However, the percent sign is a valid UNIX file name character.
- UNIX regular expressions (enclosed in brackets ([...])) are not supported.
- In some cases, specifying a file name (the portion of the path after the final slash) with a dot may be ambiguous (for example, `ls wow/my.file`).
- When the tilde (~) and slash (/) characters are specified alone, the OpenVMS system processes them as equivalent to SYS$LOGIN. However, a specification such as `~/x.x` is not valid.
- When you copy a file to or from the OpenVMS FTP server, the character string `..` (as used, for example, in the UNIX command `cd ..`) is interpreted by the OpenVMS FTP server as `[-]`. Similarly, the character string `.` is interpreted as `[]`.

Note, however, that character strings [-] and [] are not valid directory specifications on a UNIX server. These characters are interpreted by the UNIX server as file name characters.

- For the following commands, quotation marks are required around path names that start with a slash:
  - APPEND
  - CLOSE
  - DELETE
  - DIRECTORY
  - GET
  - PUT
  - VIEW
- Specifying a dot in the path name when the dot is intended to indicate the OpenVMS subdirectory can lead to indeterminate results.
- OpenVMS logical names are often specified with a trailing colon (for example, SYS$HELP:); however, this format can lead to problems with ODS-5 volumes. For example, my$topdir/subdir is interpreted as [.mytopdir.subdir]; it will not be interpreted as my$topdir:[subdir].

  To specify a logical name for a directory name, include a leading slash character and enclose the directory name in quotation marks.

1.12 Configuring User-Written Network Services

The TCP/IP Services software allows you to configure and manage network services that are not supplied with the TCP/IP Services software (user-written services). The following sections provide information about managing user-written services.

1.12.1 Starting and Stopping User-Written Services

TCP/IP Services supplies command procedures for starting and stopping user-written services. To start a user-written service, enter the following command:

```
$ SYS$STARTUP:TCPIP$CUSTOMER_SERVICE_STARTUP service-name
```

For *service-name*, specify the name of the service as defined using the TCP/IP management command SET SERVICE.

To stop the user-written service, enter the following command:

```
$ SYS$STARTUP:TCPIP$CUSTOMER_SERVICE_SHUTDOWN service-name
```

1.12.2 Specifying TCP and UDP

This section describes how to configure user-written services to use both TCP and UDP protocols. Use the TCP/IP Services management command SET SERVICE to configure the protocols for services.

You must enter a separate SET SERVICE command for each protocol. Follow these steps:

1. Set up the service using the following command:

```
$ TCPIP
TCPIP> SET SERVICE service-name /PROTOCOL=TCP -
_TCPIP> /USER_NAME=user-name /PROCESS_NAME=process -
_TCPIP> /PORT=port-number /FILE=startup-file
```

where:

- *service-name* is the name of the user-written service you are adding.
- *user-name* is an OpenVMS user account name defined in the SYSUAF file.
- *process* is the name of the service's process.
- *port-number* is the service's port number.
- *startup-file* is the service's startup file specification.

2. To add the UCP protocol, enter a second SET SERVICE command. Include the /PROTOCOL=UCP qualifier, and repeat the following required information from the first SET SERVICE command:
   - The service name (the command parameter)
   - OpenVMS user account name (the /USER qualifier)
   - The service's process name (the /PROCESS_NAME qualifier)
   - The service's port number (the /PORT qualifier)
   - The service's startup file specification (the /FILE qualifier)
   - The service's internet address, if the service is bound to a particular internet address (the /ADDRESS qualifier)

For more information about the SET SERVICE command, access online help by entering the following command:

```
TCPIP> HELP SET SERVICE
```

2

Installation, Configuration, and Startup Notes

Use this chapter in conjunction with the *Compaq TCP/IP Services for OpenVMS Installation and Configuration* manual.

2.1 Configuring IPv6

The following sections describe procedures specific to systems where IPv6 is to be enabled.

2.1.1 Information for Users of the IPv6 Early Adopter's Kit

If you are running any version of the TCP/IP Services V5.0 IPv6 early adopter's kit (EAK), remove the EAK and then install the current version of the TCP/IP Services software. You must then run the TCPIP$IP6_SETUP.COM command procedure. For more information, refer to the *Compaq TCP/IP Services for OpenVMS Guide to IPv6*.

The definition of a `sockaddr` structure has been changed. This change breaks binary compatibility for IPv6 applications that were compiled using the TCP/IP Services Version 5.0 EAK. You must recompile and relink your applications after you install the current version of TCP/IP Services.

2.1.2 Warning Message in TCPIP$CONFIG.COM

If have run the TCPIP$IP6_SETUP.COM procedure to enable IPv6, and then you run the TCPIP$CONFIG.COM command procedure, TCPIP$CONFIG.COM displays the following warning message when you select the `Core environment` option:

```
                        WARNING

This node has been configured for IPv6.  If you make any additional
changes to the configuration of the interfaces, you must run
TCPIP$IP6_SETUP again and update your host name information in
BIND/DNS for the changes to take effect.
```

2.2 Startup Problems and Restrictions

The following sections describe product startup problems.

2.2.1 Loading the Routing Database at Startup

If the BIND resolver has been configured to point only to the local host and no host name is associated with a route entry in the local hosts database, then the loading of the permanant routes database during TCP/IP Services startup fails.

To avoid this problem, define any hosts associated with the routes database in the local hosts database before you start TCP/IP Services.

2.2.2 Startup DUPLNAM Messages

When you start TCP/IP Services, the following DUPLNAM messages may appear:

```
%TCPIP-E-DYNPROXERR, cannot add record to proxy database  (TCPIP$PROXY) in dynamic memory
-SYSTEM-F-DUPLNAM, duplicate name
%TCPIP-E-DYNPROXERR, cannot add record to proxy database  (TCPIP$PROXY) in dynamic memory
-SYSTEM-F-DUPLNAM, duplicate name
%TCPIP-I-LOADSERV, loading TCPIP server proxy information
%TCPIP-I-SERVLOADED, auxiliary server loaded with 0 proxy records
-TCPIP-I-SERVSKIP, skipped 0 communication proxy records
-TCPIP-I-SERVTOTAL, total of 8 proxy records read
%TCPIP-S-STARTDONE, TCP/IP Services startup completed at  7-JUN-2000 16:03:51.48
```

You can ignore these messages. They are the result of a change in the current version of TCP/IP Services.

In previous versions of TCP/IP Services, the proxy database required that all names for a particular host be entered in the hosts database. For example, the host names `johnws` and `johnws.abc.com` needed to be in the hosts database if any NFS requests were made using either of the host names.

In the current release of TCP/IP Services, the proxy information that is loaded automatically includes all of a host's addresses and alias names. Therefore, the first entry for a host succeeds; any subsequent matching entries that differ only in the host's alias name generate DUPLNAM messages.

Proxy records for the host under multiple host names succeed, because all names (including the duplicates) are loaded.

There is only one record for each host. Therefore, if you remove a proxy entry under any of the host's names, all of the addresses and aliases for that host are removed. Subsequent removal attempts under any of the host's other names return an error.

2.3 System Page Table Entries Parameter (VAX Only)

On VAX systems, make sure the AUTOGEN parameter SPTREQ is set to at least 6000. Run SYSMAN to check the minimum SPTREQ value, as follows:

```
$ RUN SYS$SYSTEM:SYSMAN

SYSMAN> parameter show sptreq
%SYSMAN, a USE ACTIVE has been defaulted on node VMSVAX

Node VMSVAX:   Parameters in use: ACTIVE
Parameter Name    Current    Default    Minimum    Maximum Unit  Dynamic
--------------    -------    -------    -------    ------- ----  -------
SPTREQ               8000       3900       3000         -1 Pages

SYSMAN>
```

To modify the minimum SPTREQ, run the AUTOGEN command procedure as described in the *OpenVMS System Management Utilities Reference Manual: A-L*. Set the minimum SPTREQ value to 6000.

2.4 Starting the Product After a Minimum OpenVMS Boot

The product configuration and startup command procedures (TCPIP$CONFIG.COM and TCPIP$STARTUP.COM) fails if you perform any kind of boot other than a full boot. Therefore, booting OpenVMS with MIN, INST, or UPGRADE is not supported.

The TCPIP$CONFIG.COM command procedure fails on systems that do not have a SYSUAF database and a RIGHTSLIST database. These OpenVMS files must be created before you configure TCP/IP Services.

2.5 Upgrading from TCP/IP Services Version 4.*x*

The following sections describe actions you can take to preserve the behavior of the software when you upgrade from an older version of TCP/IP Services (UCX) to the current version.

2.6 Removing Prior Versions of this Product

This section provides important information you should review before installing the current version of TCP/IP Services.

2.6.1 Upgrading OpenVMS VAX Systems

The command procedure SYS$UPDATE:UCX$CLEANUP.COM is typically used to clean up a previous version of the TCP/IP Services product. However, running this command procedure when the new version of TCP/IP Services has been installed removes files that are necessary for the operation of the product.

Caution

Do not run the UCX$CLEANUP.COM command procedure after installing the new version of TCP/IP Services on an OpenVMS VAX system. If you run this command procedure, it will corrupt your TCP/IP Services installation.

Compaq strongly recommends that you remove this command procedure after installing the new version of TCP/IP Services.

2.6.2 Some UCX Files Remain After Installation

After you install and start the current version of TCP/IP Services, some files with a UCX$ prefix remain. (Most other files provided by this product use the prefix TCPIP$.) The files listed in Table 2–1 are required in order to maintain backward compatibility with previous versions of TCP/IP Services.

Table 2–1 UCX Files Required for Backward Compatibility

File	Description
SYS$LIBRARY:UCX$IPC_SHR.EXE	Allows the Compaq C Run-Time Library (CRTL) to use TCP/IP sockets.
SYS$LIBRARY:UCX$INETDEF.ADA SYS$LIBRARY:UCX$INETDEF.BAS SYS$LIBRARY:UCX$INETDEF.FOR SYS$LIBRARY:UCX$INETDEF.H SYS$LIBRARY:UCX$INETDEF.MAR SYS$LIBRARY:UCX$INETDEF.PAS SYS$LIBRARY:UCX$INETDEF.PLI SYS$LIBRARY:UCX$INETDEF.R32	The INETDEF files are shipped for compatibility with applications developed under TCP/IP Services Version 4.2. These files are identical to the files shipped with Version 4.2

.

(continued on next page)

Table 2–1 (Cont.) UCX Files Required for Backward Compatibility

File	Description
SYS$COMMON:[SYSEXE]UCX$UCP.EXE	An empty (zero block) marker file that allows some layered products that use an unsupported test for the presence of the TCP/IP Services to continue to operate.
SYS$COMMON:[SYSEXE]UCX$SERVICE.DAT	An empty (zero block) marker file may be created if the file does not exist when TCPIP$STARTUP.COM executes. The file specified by the logical name TCPIP$SERVICE (which defaults to SYS$COMMON:[SYSEXE]TCPIP$SERVICE.DAT) contains the actual service information.
SYS$STARTUP:UCX$STARTUP.COM SYS$STARTUP:UCX$CONFIG.COM	These files print an informational message to SYS$OUTPUT, then execute the corresponding TCPIP file. This allows the TCP/IP Services product to continue to operate until the system manager changes command files to use the new TCPIP prefix.
SYS$SYSTEM:UCX$LPD_SMB.EXE	Maintains backward compatibility for LPD print queues.
SYS$SHARE:UCX$ESNMP_SHR.EXE SYS$SHARE:UCX$ACCESS_SHR.EXE SYS$SHARE:UCX$RPCXDR_SHR.EXE	Shareable images required for user-written programs written under previous versions of the product.
SYS$COMMON:[SYSEXE]UCX$TELNETSYM.EXE	TELNET print symbiont executable. This file is identical to TCPIP$TELNETSYM.EXE.

2.6.3 Preserving LPD Startup and Shutdown Behavior

Your LPD startup and shutdown command procedures may contain site-specific edits. You must preserve these edits manually when you upgrade to the current version of TCP/IP Services from a previous version. The procedure for preserving your edits differs for OpenVMS Alpha systems (see Section 2.6.3.1) and OpenVMS VAX systems (see Section 2.6.3.2). To preserve your site-specific startup and shutdown command procedure files, use the procedure that is appropriate to the type of system.

2.6.3.1 Preserving LPD Behavior (Alpha Systems)

When you install TCP/IP Services over an earlier version of the product, follow the instructions displayed on your screen to preserve your edits in the LPD startup and shutdown command procedures.

The following shows a sample screen display.

```
The following product will be installed to destination:
    DEC AXPVMS TCPIP V5.1-9               DISK$ALPHASYS:[VMS$COMMON.]

UCX product already installed.

******************************************************************
Another version of TCP/IP is installed. You must execute the following
three commands before continuing with this installation:
$ BACKUP SYS$COMMON:[SYSMGR]UCX$LPD_STARTUP.COM; -
     SYS$COMMON:[SYSMGR]TCPIP$LPD_STARTUP.COM;
$ BACKUP SYS$COMMON:[SYSMGR]UCX$LPD_SHUTDOWN.COM; -
     SYS$COMMON:[SYSMGR]TCPIP$LPD_SHUTDOWN.COM;
$ PRODUCT REMOVE UCX
******************************************************************
```

After you follow these instructions and complete the installation of TCP/IP Services, your site-specific edits to the LPD startup and shutdown files are found in:

SYS$COMMON:[SYSMGR]TCPIP$LPD_STARTUP.COM_OLD
SYS$COMMON:[SYSMGR]TCPIP$LPD_SHUTDOWN.COM_OLD

Now merge your site-specific edits into:

SYS$COMMON:[SYSMGR]TCPIP$LPD_SYSTARTUP.COM
SYS$COMMON:[SYSMGR]TCPIP$LPD_SYSHUTDOWN.COM

2.6.3.2 Preserving LPD Behavior (VAX Systems)

To preserve your site-specific startup and shutdown information, you must install TCP/IP Services, then copy the site-specific edits from:

SYS$COMMON:[SYSMGR]UCX$LPD_STARTUP.COM
SYS$COMMON:[SYSMGR]UCX$LPD_SHUTDOWN.COM

to the following files:

SYS$COMMON:[SYSMGR]TCPIP$LPD_STARTUP.COM
SYS$COMMON:[SYSMGR]TCPIP$LPD_SHUTDOWN.COM

2.6.3.3 Merging Edits (Alpha and VAX Systems)

When you merge edits, do not include the commands to start and stop the queue UCX$LPD_QUEUE. This queue has been replaced with TCPIP$LPD_QUEUE. The commands for starting and stopping TCPIP$LPD_QUEUE are in the LPD startup and shutdown command procedure files.

After you merge the edits, modify the value of the /PROCESSOR qualifier in the LPD client queue startup commands that you have just appended, replacing TCPIP$LPD_SMB with UCX$LPD_SMB. For example, enter the following command:

```
LSE Command> SUBSTITUTE/ALL "ucx$lpd_smb" "tcpip$lpd_smb"
```

2.6.4 Saving Mail Messages When You Upgrade

The new version of SMTP includes control files that are different from previous versions. Before upgrading to the current version of TCP/IP Services, run the ANALYZE utility to pick up any dead letters (SMTP control files that have not been submitted to a print queue), as follows:

```
$ ANALYZE MAIL/REPAIR
```

2.6.5 Preserving SNMP Startup and Shutdown Behavior

After you upgrade to the current version of TCP/IP Services, you must perform one of the following actions to ensure correct SNMP startup:

- If SNMP was configured under an old TCP/IP Services installation (UCX) and you want to retain the previous configuration, run the SYS$MANAGER:TCPIP$CONFIG.COM command procedure and select the option to automatically convert UCX configuration files.
- After you upgrade to the current version of TCP/IP Services, run the SYS$MANAGER:TCPIP$CONFIG.COM command procedure. If SNMP is still enabled, disable SNMP then enable it again. This is necessary for the proper operation of this component.

If you have customized versions of the UCX$SNMP_STARTUP.COM and UCX$SNMP_SHUTDOWN.COM command procedures (used to start and stop extension subagents), save your customized files to a different directory before upgrading to the new version of TCP/IP Services. If you do not perform this step, your customized changes will be lost.

Check for versions of these files in the following locations:

- SYS$MANAGER
- SYS$STARTUP
- SYS$SYSDEVICE:[UCX$SNMP]

After you install TCP/IP Services, manually merge your saved changes into the new files created after installation. For more information, see the *Compaq TCP/IP Services for OpenVMS Management* manual.

2.7 SNMP Installation and Setup Notes

The following sections describe procedures for installing and setting up SNMP.

2.7.1 SNMP Messages When You Install TCP/IP Services

For sites where the same version of TCP/IP Services is installed multiple times, informational messages similar to the following may appear in the installation dialog:

```
Do you want to review the options? [NO]

Execution phase starting ...

The following product will be installed to destination:
    DEC AXPVMS TCPIP T5.3-9I              DISK$AXPVMSSYS:[VMS$COMMON.]
The following product will be removed from destination:
    DEC AXPVMS TCPIP T5.3-9H              DISK$AXPVMSSYS:[VMS$COMMON.]
%PCSI-I-RETAIN, file [SYSEXE]TCPIP$ESNMP_SERVER.EXE was not replaced because
file from kit does not have higher generation number
%PCSI-I-RETAIN, file [SYSEXE]TCPIP$HR_MIB.EXE was not replaced because file
from kit does not have higher generation number
%PCSI-I-RETAIN, file [SYSEXE]TCPIP$OS_MIBS.EXE was not replaced because file
from kit does not have higher generation number
%PCSI-I-RETAIN, file [SYSLIB]TCPIP$ESNMP_SHR.EXE was not replaced because file
from kit does not have higher generation number
%PCSI-I-RETAIN, file [SYSLIB]UCX$ESNMP_SHR.EXE was not replaced because file
from kit does not have higher generation number
```

You can ignore these messages.

2.7.2 Verifying the SNMP Installation

SNMP has a separate installation verification procedure (IVP). To verify your SNMP configuration, perform these steps:

1. Configure the SNMP services using the TCPIP$CONFIG.COM command procedure, and start the TCP/IP Services software. Make sure that your process has SYSTEM user privileges.
2. Run the TCP/IP Services configuration command procedure by entering the following command:

   ```
   $ @SYS$MANAGER:TCPIP$CONFIG
   ```

3. Choose option 7 (Run tests).

4. From the Compaq TCP/IP Services for OpenVMS TEST Menu, choose option 2.
5. To run the SNMP IVP any time after exiting the configuration procedure, enter the following command:

```
$ RUN SYS$COMMON:[SYSTEST.TCPIP]TCPIP$SNMPIVP.EXE
```

2.7.3 SNMP Subagent Startup Messages

The SNMP startup procedure can produce the following error messages in subagent log files:

```
25-JUL-2001 14:13:32.47 **ERROR ESNMP_INIT.C line 3777: Could not
connect to master: connection refused
25-JUL-2001 14:13:32.94 WARNING OS_MIBS.C line 942: Master agent
cannot be reached.  Waiting to attempt reconnect.
```

These messages are the result of a timing problem and can be ignored.

2.8 Setting Up the TCP/IP Services Message Database

At installation, the TCP/IP Services message database is installed at SYS$COMMON:[SYSHLP]TCPIP.MSGHLP$DATA.

To get information about TCP/IP messages, include this database with the OpenVMS message database, as follows:

1. Define the logical name MSGHLP$LIBRARY to point to all the databases in the directory:

```
$ DEFINE/SYSTEM MSGHLP$LIBRARY SYS$COMMON:[SYSHLP]*.MSGHLP$DATA
```

2. Enter the DCL command HELP/MESSAGE to make sure the TCP/IP message database is now recognized. For example:

```
$ HELP/MESSAGE FTP

 SESDCN,  FTPD: Session disconnection from 'host' at 'time'

  Facility:     TCPIP, FTP Server

  Explanation:  This message appears when a session is disconnected, stating
                the name of the client initiating the disconnection and the
                time of the disconnection.

  User Action:  None.

Press RETURN to continue ...
```

In this example, Help Message displays all the messages that contain FTP as part of the message ID.

2.9 Troubleshooting SMTP and LPD Shutdown Problems

If SMTP or LPD shutdown generates errors indicating that the queue manager is not running, check your site-specific shutdown command procedure (VMS_SYSHUTDOWN.COM). If this procedure contains the command to stop the queue manager (STOP/QUEUE/MANAGER), make sure this command comes after any invocation of the TCPIP$SHUTDOWN.COM procedure.

Note

You do not have to stop the queue manager explicitly. The queue manager is automatically stopped and started when you restart the system.

3
Problems and Restrictions

This chapter provides information about problems and restrictions in the current version of TCP/IP Services.

3.1 Determining the TCP/IP Device Name from a Channel Assignment

OpenVMS provides several ways to determine the name of a device on a channel assignment. Using the SYS$GETDVI/SYS$GETDVIW system services, the DVI$_DEVNAM, DVI$_FULLDEVNAM, and DVI$_UNIT items all return information about the device. While the first two provide the full device name, DVI$_UNIT returns only the unit number of the device. To form the complete device name, a program must prefix the unit number (as a string) with the device name and controller information. In the case of the TCP/IP device name, the programmer could add the string `BG` or `BGA`. For example, `BG + 1234` would produce the device name `BG1234:`.

The TCP/IP device name may be altered in a future release. It is good programming practice to use the DVI$_DEVNAM or DVI$_FULLDEVNAM items to obtain the full device-name string. Such programs are not based on the assumption that the TCP/IP device name is BG*nnnn* or BGA*nnnn*, and would not be affected by any change in the TCP/IP device name strategy.

3.2 RCP Full Transparent Copy Operations

RCP on OpenVMS is best used for transferring text files. By default, RCP converts any type of OpenVMS file that is not Stream_LF to Stream_LF format using the standard OpenVMS $CONVERT utility with the following conversion specification:

```
FILE;ORGA SEQU;RECO;CARR CARR;FORM STREAM_LF;SIZE 0;BLOCK YES
```

Then RCP sends the converted file using block-mode RMS file I/O (SYS$READ()) and on receive writes the data using block-mode (SYS$WRITE()).

This behavior has been changed so that RCP does not convert Fixed or Undefined files (in addition to Stream_LF files). You can restore the old behavior using with the TCPIP$RCP_SEND_FIX_FORMAT_AS_ASCII logical name. If this logical name is set, the original behavior of converting Fixed and Undefined files is restored. If this logical is set to a number other than 1, the original behavior is restored, except for files with a fixed-length record size that exactly matches the value of this logical name, which are not converted.

For example, if you define this logical to 512, all Fixed and Undefined files are converted except for Fixed files with a fixed-length record size of 512 (such as OpenVMS executable image files).

The receiving peer, if OpenVMS, always creates a file of type Stream_LF. The RCP protocol provides no method of transferring file type information between sender and receiver. Therefore, the receiving peer has no way of knowing anything about file structure.

In an OpenVMS-to-OpenVMS transfer, if the original file was Fixed or Undefined and was not converted, the user can change the attributes on the Stream_LF copy to correspond to the format of the original file. This can be accomplished using the DCL command SET FILE/ATTRIBUTES.

For example, after transferring an OpenVMS executable image file (Fixed with a record-length of 512 bytes), enter the following command to make it an executable again:

```
$ SET FILE/ATTR=(RFM:FIX,LRL:512) RCP-COPIED-FILE.EXE
```

RCP also has file-size limitations. These are due to RCP's dependence on the Compaq C RTL (run-time library). The RCP protocol requires the length of the file to be sent as part of the protocol. The length is interpreted as a signed 32-bit integer. On OpenVMS, the file's length is determined using a Compaq C RTL call to `fstat()`. Therefore, files transferred using RCP must be less than 2 GB minus 1 byte (2147483647 bytes).

In comparison, FTP does not have any of these limitations, but it utilizes a different security model.

3.3 BIND Version 9 Does Not Run on VAX Systems

The new BIND Version 9 DNS server does not run on VAX systems. Please note that future support of BIND 8 on VAX systems will be limited. If you are running a BIND server on a VAX system, you should upgrade to an Alpha system.

3.4 NFS Problems and Restrictions

The following sections describe problems and restrictions with NFS.

3.4.1 NFS Server Problems and Restrictions

If the NFS server and the NFS client are in different domains and unqualified host names are used in requests, the lock server (LOCKD) fails to honor the request and leaves the file unlocked.

When the server attempts to look up a host using its unqualified host name (for example, `johnws`) instead of the fully qualified host name (for example, `johnws.abc com`), and the host is not in the same domain as the server, the request fails.

To solve this type of problem, you can do one of the following:

- When you configure the NFS client, specify the fully qualified host name, including the domain name. This ensures that translation will succeed.
- Add an entry to the NFS server's hosts database for the client's unqualified host name. Only that NFS server will be able to translate this host name. This solution will not work if the client obtains its address dynamically from DHCP.

3.4.2 NFS Client Problems and Restrictions

- To get proper timestamps, when the system time is changed for daylight savings time (DST), dismount all DNFS devices. (The TCP/IP management command SHOW MOUNT should show zero mounted devices.) Then remount the devices.
- The NFS client should properly handle file names with the semicolon character on ODS-5 disk volumes. (For example, `a^;b.dat;5` is a valid file name.)

  The current version does not handle these types of file names properly; they are truncated at the semicolon.
- The NFS client included with TCP/IP Services uses the NFS Version 2 protocol only.
- With the NFS Version 2 protocol, the value of the file size is limited to 32 bits.
- The ISO Latin-1 character set is supported. The UCS-2 characters are not supported.
- File names, including file extensions, can be no more than 236 characters long.
- Files containing characters not accepted by ODS-5 on the active OpenVMS version or whose name and extension exceeds 236 characters are truncated to zero length. This makes them invisible to OpenVMS and is consistent with prior OpenVMS NFS client behavior.

3.5 IPv6 Requires the BIND Resolver

If you are using IPv6, you must enable the BIND resolver. If you do not have the BIND resolver configured, you can enable it using the TCPIP$CONFIG.COM command procedure. From the Core menu, select BIND Resolver. If you do not have access to a BIND server, specify the node address 127.0.0.0 as your BIND server. You must specify the BIND server to enable the BIND resolver.

3.6 TCP/IP Management Command Restrictions

The following restrictions apply to the TCP/IP management commands:

- SET NAME_SERVICE /PATH

  This command requires the SYSNAM privilege. If you enter the command without the appropriate privilege at the process level, the command does not work and you are not notified. If you enter the command at the system level, the command does not work but you do receive an error message.
- SET SERVICE command

  When you modify parameters to a service, disable and reenable the service for the modifications to take effect.

3.7 NTP Problems and Restrictions

The NTP server has a stratum limit of 15. The server does not synchronize to any time server that reports a stratum of 15 or greater. This may cause problems if you try to synchronize to a server running the UCX NTP server, if that server has been designated as "free running" (with the `local-master` command). For proper operation, the `local-master` designation must be specified with a stratum no greater than 14.

3.8 SNMP Problems

This section describes restrictions to the SNMP component for this release.

3.8.1 Incomplete Restart

When the SNMP master and subagents fail or are stopped, TCP/IP Services is often able to restart all processes automatically. However, under certain conditions, subagent processes may not restart; that is, the DCL command SHOW SYSTEM display does not include TCPIP$OS_MIBS and TCPIP$HR_MIB. If this situation occurs, restart SNMP by entering the following commands:

```
$ @SYS$STARTUP:TCPIP$SNMP_SHUTDOWN
$ @SYS$STARTUP:TCPIP$SNMP_STARTUP
```

3.8.2 SNMP IVP Error

On slow systems, the SNMP Installation Verification Procedure can fail because a subagent does not respond to the test query. The error messages look like this:

```
. . .
Shutting down the SNMP service... done.


Creating temporary read/write community SNMPIVP_153.

Enabling SET operations.

Starting the SNMP service... done.

SNMPIVP: unexpected text in response to SNMP request:
"- no such name - returned for variable 1"
See file SYS$SYSDEVICE:[TCPIP$SNMP]TCPIP$SNMP_REQUEST.DAT for more
details.
sysContact could not be retrieved.  Status = 0
The SNMP IVP has NOT completed successfully.
SNMP IVP request completed.
Press Return to continue ...
```

These types of messages in the IVP can be safely ignored.

3.8.3 Using Existing MIB Subagent Modules

If an existing subagent does not execute properly, you may need to relink it against the current version of TCP/IP Services to produce a working image. Some subagents (such as those for OpenVMS support of Compaq Insight Manager) also require a minimum version of OpenVMS and a minimum version of TCP/IP Services.

The following general restrictions and cautions apply:

- In general, only executable images linked against the following versions of the eSNMP shareable image are upward compatible with the current version of TCP/IP Services:
  - UCX$ESNMP_SHR.EXE from TCP/IP Services Version 4.2 ECO 4

- TCPIP$ESNMP_SHR.EXE from TCP/IP Services Version 5.0A ECO 1

Images built under versions other than these can be relinked with one of the shareable images, or with TCPIP$ESNMP_SHR.EXE in the current version of TCP/IP Services.

- The underlying eSNMP API changed from DPI in Version 5.0 to AgentX in the current version of TCP/IP Services. Therefore, executable images linked against older object library versions of the API (*$ESNMP.OLB) must be relinked against either the new object library or the new shareable image. Linking against the shareable image ensures future upward compatibility and results in smaller image sizes.

Note

Although images can run without being relinked, backward compatibility is not guaranteed. These images can result in inaccurate data or run-time problems.

- Programs that rely on TCP/IP Services Version 4.2 kernel data structures or functions may run but may not return valid data. Such programs should be rewritten.
- Executable images linked against UCX$ACCESS_SHR.EXE, UCX$IPC_SHR.EXE, UCX$RPCXDR_SHR.EXE, or other older shareable images, may not run even when relinked. You may need to recompile, or rewrite and recompile, such programs.

  If you have executable images linked against versions other than those listed above and cannot relink, contact your Compaq support representative for assistance.
- If you have problems running executable images linked against TCP/IP Services Version 4.2 ECO 4 or TCP/IP Services Version 5.0A ECO 1, verify that the version of the shareable image is the latest by entering the following DCL command:

```
$ DIRECTORY/DATE/PROTECTION SYS$SHARE:*$ESNMP_SHR.EXE
```

  The creation dates of the files with the prefix TCPIP$ and UCX$ should be within a few seconds of each other, and only one version of each file should exist. Make sure both images include the file protection W:RE.

  Also, you can use the following command to check the version:

```
$ TCPIP SHOW VERSION/ALL
```

  The second column in the line for image TCPIP$ESNMP_SHR under the category “Network Management” displays the version. For example:

```
TCPIP$ESNMP_SHR;1      V5.1-15B      23-MAY-2001  SYS$COMMON:[SYSLIB]
```

- Both TCP/IP Services Version 5.0A ECO 1 and TCP/IP Services Version 5.1 provide an updated version of the UCX$ESNMP_SHR.EXE shareable image to provide compatibility with subagents linked under TCP/IP Services Version 4.2 ECO 4. Do not delete this file.

3.8.4 Restrictions to RFC-Defined Functionality

- SNMP requests are not implemented for the following MIB-II group objects:

  ```
  ipRouteMetric1 - ipRouteMetric5
  tcpMaxConn
  ```

- SNMP requests are not implemented for the following Host Resources MIB objects:

  ```
  hrPartitionTable
  hrPrinterTable
  hrSWInstalled
  hrSWInstalledTable
  ```

- SNMP `set` requests are not implemented for the following MIB-II group objects:

  ```
  ipDefaultTTL
  ipRouteAge
  ipRouteDest
  ipRouteIfIndex
  ipRouteMask
  ipRouteNextHop
  ipRouteType
  ```

- SNMP `set` requests are not implemented for the following Host Resources MIB objects:

  ```
  hrFSLastFullBackupDate
  hrFSLastPartialBackupDate
  hrStorageSize
  hrSWRunStatus
  hrSystemDate
  hrSystemInitialLoadDevice
  hrSystemInitialLoadParameters
  ```

- In the SNMP group (1.3.6.1.2.1.11), data elements noted as obsolete in RFC 1907 are not implemented.

3.8.5 SNMP Restrictions and Characteristics

This section describes restrictions and characteristics of SNMP. For more information, refer to the *Compaq TCP/IP Services for OpenVMS SNMP Programming and Reference* guide.

- The SNMP server responds correctly to SNMP requests directed to a cluster alias. Note, however, that an unexpected host might be reached when querying from a TCP/IP Services Version 4.*x* system that is a member of a cluster group but is not the current impersonator.
- The SNMP master agent and subagents do not start if the value of logical name TCPIP$INET_HOST does not yield the IP address of a functional interface on the host when used in a DNS query. This problem does not occur if the server host is configured correctly with a permanent network connection (for example, Ethernet or FDDI). The problem can occur when a host is connected through PPP and the IP address used for the PPP connection does not match the IP address of the TCPIP$INET_HOST logical name.
- Under certain conditions observed primarily on OpenVMS VAX systems, the master agent or subagent exits with an error from an internal `select()` socket call. In most circumstances, looping does not occur.

You can control the number of iterations if looping occurs by defining the TCPIP$SNMP_SELECT_ERROR_LIMIT logical name.

- The MIB browser provided with TCP/IP Services (TCPIP$SNMP_REQUEST.EXE) supports getnext processing of OIDs that include the 32-bit OpenVMS process ID as a component. However, other MIB browsers might not provide this support.

  For example, the following OIDs and values are supported on OpenVMS:

  ```
  1.3.6.1.2.1.25.4.2.1.1.1321206828 = 1321206828
  1.3.6.1.2.1.25.4.2.1.1.1321206829 = 1321206829
  1.3.6.1.2.1.25.4.2.1.1.1321206830 = 1321206830
  ```

  These examples are from hrSWRunTable; the hrSWRunPerfTable might be affected as well.

- sysObjectID is returned in the following format, as required for interoperability with Compaq Insight Manager/XE:

  ```
  1.3.6.1.2.1.1.2.0 = 1.3.6.1.4.1.36.2.15.22.1
  ```

  In this format, the right-hand value corresponds to:

```
iso.org.dod.internet.private.enterprises.dec.ema.sysObjectIds.DEC-OpenVMS.eSNMP
```

- The sysORTable is implemented (see RFC 1907 for details). Elements are under OID prefix 1.3.6.1.2.1.1.9.1.

  When both the TCPIP$OS_MIBS and TCPIP$HR_MIB subagents are running, a get request on the sysORTable is as follows. Except where noted, the OIDs conform to RFC 1907:

  ```
  1.3.6.1.2.1.1.9.1.2.1 = 1.3.6.1.4.1.36.15.3.3.1.1
  1.3.6.1.2.1.1.9.1.2.2 = 1.3.6.1.4.1.36.15.3.3.1.2
  1.3.6.1.2.1.1.9.1.3.1 = Base o/s agent (OS_MIBS) capabilities
  1.3.6.1.2.1.1.9.1.3.2 = Base o/s agent (HR_MIB) capabilities
  1.3.6.1.2.1.1.9.1.4.1 = 31 = 0 d 0:0:0
  1.3.6.1.2.1.1.9.1.4.2 = 36 = 0 d 0:0:0
  ```

  This example is from the MIB browser (TCPIP$SNMP_REQUEST.EXE).

- The hrDeviceTable now includes template devices (for example, DNFS0 for NFS and DAD0 for virtual devices).
- For network devices, only the template devices (those with unit number 0) are displayed.
- The hrDeviceTable includes all devices known to the OpenVMS host except those with the following characteristics.
  - Off-line
  - Remote
  - UCB marked delete-on-zero-reference-count
  - Mailbox device
  - Device with remote terminal (DEV$M_RTT characteristic)
  - Template terminal-class device
  - LAT device (begins with _LT)
  - Virtual terminal device (begins with _VT)
  - Pseudoterminal device (begins with _FT)

- Data items in the hrDevTable group have the following restrictions:
  - hrDeviceID always contains a null OID (0.0).
  - hrDeviceErrors is coded as described in the following table.

Code	Condition
warning (3)	If error logging is in progress (OpenVMS UCB value UCB$M_ERLOGIP).
running (2)	If status indicates software is valid and no error logging in progress (OpenVMS UCB value UCB$M_VALID).
unknown (1)	Any other OpenVMS status.

- On systems running versions of the operating system prior to OpenVMS 7.1-2, counters for the MIB-II ifTable do not wrap back to zero after reaching the maximum value defined in RFC 1155 ($2^{32} - 1$). Instead, they behave like Gauge counters and remain at the maximum value until cleared by an external event such as a system reboot. The counters affected are as follows:

```
ifInDiscards
ifInErrors
ifInNUcastPkts
ifInOctets
ifInUcastPkts
ifInUnknownProtos
ifOutErrors
ifOutNUcastPkts
ifOutOctets
ifOutUcastPkts
```

  Note that for SNMPv2, these counters are data type Counter32. The following ifTable members are always -1 for OpenVMS: ifOutDiscards (Counter32) and ifOutQLen (Gauge32).

- Under certain conditions a subagent name may be listed more than once in the sysORTable. For example:

```
1.3.6.1.2.1.1.9.1.2.2 = 1.3.6.1.4.1.36.2.15.22.1.1.1
1.3.6.1.2.1.1.9.1.2.3 = 1.3.6.1.4.1.36.2.15.22.1.1.1
1.3.6.1.2.1.1.9.1.3.2 = Base o/s agent (OS_MIBS) capabilities
1.3.6.1.2.1.1.9.1.3.3 = Base o/s agent (OS_MIBS) capabilities
1.3.6.1.2.1.1.9.1.4.2 = 11351 = 0 d 0:1:53
1.3.6.1.2.1.1.9.1.4.3 = 17829 = 0 d 0:2:58
```

  In this example, the OS_MIBS subagent is listed twice because two instances of the image SYS$SYSTEM:TCPIP$OS_MIBS.EXE are running. (This condition indicates a problem with TCP/IP Services.) The text values for OIDs with the following prefix are sent by subagents, and do not reflect an error in the SNMP client:

```
1.3.6.1.2.1.1.9.1.3
```

- hrFSMountPoint (1.3.6.1.2.1.25.3.8.1.2) is DNFS*n*. The device may change between restarts or after a dismount/mount procedure.
- In the hrFSTable group, if no file systems are mounted through NFS or no information is accessible, a "no such instance" status is returned for a get request. Browsers respond differently to this message. For example, TCPIP$SNMP_REQUEST.EXE responds with no output and returns directly to the DCL prompt.

After an NFS mount, the following information is returned in response to a get request. The data items implemented for OpenVMS (refer to RFC 1514) are:

- hrFSIndex
- hrFSMountPoint: local DNFS device name.
- hrFSRemoteMountPoint: In UNIX format, the remote file system name.
- hrFSType: OID 1.3.6.1.2.1.25.3.9.1, if the remote system is running TCP/IP Services and the file system is not a container file system. hrFSNFS, OID 1.3.6.1.2.1.25.3.9.14, if OpenVMS TCP/IP container file system or UNIX host.
- hrFSAccess (as defined in RFC 1514).
- hrFSBootable: always HRM_FALSE (integer 2).
- hrFSStorageIndex: always 0.
- hrFSLastFullBackupDate: unknown time. Encoded according to RFC 1514 as hexadecimal value 00 00 01 01 00 00 00 00 (January 1, 0000).
- hrFSLastPartialBackupDate: unknown time. Not available for OpenVMS systems. Instead, the hexadecimal value 00-00-01-01-00-00-00-00 (January 1, 0000) applies.

- hrProcessorFrwID (OID prefix 1.3.6.1.2.1.25.3.3.1.1):
  - Not implemented on OpenVMS VAX. Always returns standard null OID (0.0).

    ```
    1.3.6.1.2.1.25.3.3.1.1.1 = 0.0
    ```

  - An example for OpenVMS Alpha follows. The example corresponds to firmware version 5.56-7.

    ```
    1.3.6.1.2.1.25.3.3.1.1.1 = 1.3.6.1.2.1.25.3.3.1.1.1.5.56.7
    ```

- Data items in the hrDiskStorage table have the following restrictions:
  - hrDiskStorageMedia: Always "unknown" (2).
  - hrDiskStorageRemoveble: Always "false" (2). Note the incorrect spelling of Removeable in hrDiskStorageRemoveble (from RFC 1514).
- hrStorageType always contain the value of hrStorageFixedDisk (1.3.6.1.2.1.25.2.1.4).
- You can ignore the following warning that appears in the log file if a null OID value (0.0) for an instance is retrieved in response to a Get, GetNext, or GetBulk request:

  ```
  o_oid; Null oid or oid->elements, or oid->nelem == 0
  ```

3.8.6 Upgrading SNMP

After upgrading to the current version of TCP/IP Services, you must disable and then enable SNMP using the TCPIP$CONFIG configuration command procedure. When prompted for “this node” or “all nodes,” select the option that reflects the previous configuration.

3.8.7 Communication Controller Data Not Fully Updated

When you upgrade TCP/IP Services and then modify an existing communication controller, programs that use the communication controller might not have access to the updated information.

To ensure that programs like the MIB browser (SNMP_REQUEST) have access to the new data about the communication controller, do the following:

1. Delete the communication controller using the TCP/IP management command DELETE COMMUNICATION_CONTROLLER.
2. Reset the communication controller by running the TCPIP$CONFIG.COM command procedure and exiting.
3. Restart the program (such as SNMP) by entering the following commands:

   ```
   $ @SYS$STARTUP:SNMP_SHUTDOWN.COM

   $ @SYS$STARTUP:SNMP_STARTUP.COM
   ```

4. Use the TCP/IP management command LIST COMMUNICATION_CONTROLLER to display the information.

3.8.8 SNMP MIB Browser Usage

If you use either the `-l` (loop mode) or `-t` (tree mode) flag, you cannot also specify the `-m` (maximum repetitions) flag or the `-n` (nonrepeaters) flag. The latter flags are incompatible with loop mode and tree mode.

Incorrect use of the `-n` and `-m` flags results in the following messages:

```
$ snmp_request mynode.co.com public getbulk -v2c -n 20 -m 10 -t 1.3.6.1.2.1
Warning: -n reset to 0 since -l or -t flag is specified.
Warning: -m reset to 1 since -l or -t flag is specified.
1.3.6.1.2.1.1.1.0 = mynode.company.com
```

3.8.9 Duplicate Subagent Identifiers

With this version of TCP/IP Services, two subagents can have the same identifier parameter. Be aware, however, that having two subagents with the same name makes it difficult to determine the cause of problems reported in the log file.

3.8.10 eSNMP Programming and Subagent Development

The following notes pertain to eSNMP programming and subagent development.

- In the documentation, the terms **extension subagent**, **custom subagent**, and **user-written subagent** refer to any subagent other than the standard subagents for MIB-II and the Host Resources MIB, which are provided as part of the TCP/IP Services product.
- In the [.SNMP] subdirectory of TCPIP$EXAMPLES, files with the .C, .H, .COM, .MY, and .AWK extensions contain additional comments and documentation.
- The TCPIP$SNMP_REQUEST.EXE, TCPIP$SNMP_TRAPSND.EXE, and TCPIP$SNMP_TRAPSND.EXE programs are useful for testing during extension subagent development.
- For information about prototypes and definitions for the routines in the eSNMP API, see the TCPIP$SNMP:ESNMP.H file.

4
Corrections

This chapter describes the problems corrected in this version of TCP/IP Services. It contains the following sections:

- Section 4.1 describes software corrections in this release.
- Section 4.2 describes customer-reported problems corrected in this release.

4.1 Software Corrections

The following sections describe the software corrections included in this release of TCP/IP Services, organized by component.

4.1.1 Management Command Interface Problems Fixed in This Release

- The ROUTE command no longer fails when the interface name is entered in lowercase letters. It is no longer necessary to put quotation marks around the interface name. The interface name is always accepted and is handled as uppercase.
- If Ctrl/C was issued at the right moment during a SHOW HOST/NOLOCAL command, an ACCVIO error could occur.

  Also, some `errno` status returns were being passed back to the management command interface, which was unable to translate them.
- The management command interface failed to consistently handle UNIX management utility commands, whether launched as a DCL command or TCP/IP management command. Sometimes it was necessary to quote lowercase characters. At other times, quotation marks were not necessary.

  When you enter DCL commands to manage TCP/IP Services, it is no longer necessary to include quotation marks around UNIX utility commands (for example, `TCPIP "ifconfig -a"`). You need to enclose only uppercase options in quotation marks (for example, `TCPIP ifconfig "-Q" inet`).
- The following TCP/IP management commands have been enhanced to support BIND Version 9 (Alpha systems only):
  - The SET NAME /INITIALIZE command now reloads all BIND databases and the BIND configuration file. It requires that at least one of the following process privileges be set: BYPASS, READALL, or SYSPRV. In addition, it also requires that either TCPIP$ETC:RNDC.CONF or TCPIP$ETC:RNDC.KEY be set up to allow for secure communication between the user and the BIND server.
  - The SHOW NAME /STATISTICS command now writes statistics to the SYS$SPECIFIC:[TCPIP$BIND]TCPIP$BIND.STATS file. The command requires that at least one of the following process privileges be set: BYPASS, READALL, or SYSPRV. In addition, it also requires that either

TCPIP$ETC:RNDC.CONF or TCPIP$ETC:RNDC.KEY be set up to allow for secure communication between the user and the BIND server.

4.1.2 BIND Problems Fixed in This Release

- The ndc commands start, stop, and restart were broken. This correction is relevant only to VAX systems. On Alpha systems, the ndc utility has been replaced by the rndc utility.
- In some cases, with a search path defined, the nslookup utility and the TCP/IP management command SHOW HOST returned the "host-not-found" message before it exhaustsed the entire search path list.
- TCPIP$BINDSETUP.COM did not restart BIND properly.

4.1.3 BIND Resolver Problems Fixed in This Release

If the BIND resolver was not enabled, applications that used IP addresses resulted in errors if the address was not defined in the local hosts database.

4.1.4 IPC Problems Fixed in This Release

The send()) and recv()) functions did not allow the use of 64-bit addresses.

The send()) and recv()) functions now allow 64-bit addresses. Note that the sendto()), sendmsg()), recvfrom()), and recvmsg()) functions support only 32-bit addresses.

This change affects Alpha systems only.

4.1.5 SMTP Problems Fixed in This Release

TCP/IP Services and OpenVMS version information was revealed by the SMTP software. This may be a security issue.

The information was revealed in the following places:

- The SMTP server's response to the initial connection from the SMTP client.
- The Received header appended to the list of Received headers by the SMTP server.
- The Received header inserted into the list of Received headers by the Send-From-File feature.

To tells the SMTP software not to reveal TCP/IP Services or OpenVMS version information, set the TCPIP$SMTP_SUPPRESS_VERSION_INFO system logical name.

To define the logical, enter the following command:

```
$ DEFINE /SYSTEM TCPIP$SMTP_SUPPRESS_VERSION_INFO
```

4.1.6 SNMP Problems Fixed in This Release

The SNMP protocol has been updated with the following corrections:

- This version of TCP/IP Services supports the AGENTX MIB. The OID root for this new MIB is:

  ```
  1.3.6.1.2.1.74
  ```

- The Chess example is changed to use the standard OpenVMS `sysObjectID`, to match the one used for Compaq Insight Manager compatibility in the `OS` and `HR` MIBs. The numeric value is:

  ```
  1.3.6.1.4.1.36.2.15.22.1.
  ```

- False duplicate error messages when the subagent image does not exit but when the subagent needs to restart (because of loss of contact with master agent) are eliminated.
- Support is added for 32-bit subidentifiers in numeric OIDs.
- Traps are now generated when the client makes a SET request to a server using a valid community name but when the client does not have write permission for the client host.
- A configuration option for maximum SNMP message size has been added. The following logical names are available for restricting the size of SNMP packet size allowed by the master agent:
  - SNMP_MAX_GETSET_LEN, which specifies the size allowed from incoming GET, GETNEXT, GETBULK and SET requests.
  - SNMP_MAX_MSG_LEN, which specifies the size allowed from incoming GET, GETNEXT, GETBULK and SET requests, and also the size allowed for outgoing trap messages.

  To control these sizes, use the following DCL commands interactively or in a system startup file:

  ```
  $ ASSIGN /SYSTEM /EXEC size TCPIP$SNMP_MAX_GETSET_LEN
  $ ASSIGN /SYSTEM /EXEC size TCPIP$SNMP_MAX_MSG_LEN
  ```

  Compaq recommends that any changes to these sizes be made with extreme caution. The values specified for *size* should be between 484 and 1500.
- On systems with a large number of ARP entries or TCP or UDP connections (on the order of 10,000), the MIB-II returns the following error message:

  ```
  Exceeded maximum kinfo call level (1000)
  ```

4.1.7 FTP Problems Fixed in this Release

The FTP program is updated with the following corrections:

- The FTP commands `ls` and `directory` did not display files recursively. (That is, the contents of subdirectories were not displayed.)

  To activate recursive displays of directory contents, define the following logical name at the system level:

  ```
  $ ASSIGN/SYSTEM 1 TCPIP$FTPD_DIR_RECURSIVE
  ```

  On OpenVMS Version 7.2, if the file specification includes the path name preceding the file name and also contains a wildcard character, the directory listing is recursive.

  The top-level directory is always included in the display except when input file names do not contain a path, include a wildcard character, and end with a backslash (\) character.

- On systems with Version 7.2 of the Compaq C shareable library (DECC$SHR.EXE), the following commands failed to set the local default if the subdirectory had been assigned a logical name:

```
$ SHOW LOGICAL NEW
   "NEW" = "MYNEW" (LNM$PROCESS_TABLE)

$ FTP
FTP> lcd new
%TCPIP-E-FTP_NOSUCHFILE, no such file new:
%Failed to set default directory to [.new:]
Local directory now SYS$SYSROOT:[SYSMGR]
```

4.2 Reported Problems Corrected in this Release

Compaq uses the Integrated Problem Management Tool (IPMT) to track problems reported by customers. When a problem report is created, it is given a unique identification number starting with "CFS." When the problem is logged for engineering attention, it receives a Problem Tracking Report (PTR) number.

This section describes problems that have been reported and corrected in this release of TCP/IP Services, including the CFS and PTR numbers for each.

Table 4–1 describes the customer-reported problems that have been corrected in the BIND resolver.

Table 4–1 BIND Resolver Problems Fixed in this Release

CFS Number	PTR Number	Description
CFS.82606	70-5-1641	Previously, applications such as the FTP client and TELNET client may have encountered a BIND resolver restriction that did not allow host names to contain underscore characters (_). The applications returned either an "unexpected nameserver" error or a "nameserver experienced temporary error, retry operation" error. This problem has been fixed. The BIND resolver recognizes underscores in host names. Note that the BIND server can still restrict the characters that are allowed in host names.

Table 4–2 LBROKER Problems Fixed in this Release

CFS Number	PTR Number	Description
CFS.80807	70-5-1572	The dynamic updates being sent by the load broker to a Windows 2000 DNS server failed with an error similiar to `unknown response: ans=0, auth=1, add=1, rcode=3`. This problem has been corrected.
CFS.84739	70-5-1729	The Load Broker failed during startup. This problem is corrected. The maximum number of NS records per zone has been increased from 16 to 32.

Table 4–3 TELNET Problems Corrected in this Release

CFS Number	PTR Number	Description
CFS.73400 CFS.75491 CFS.75696	70-5-1260 70-5-1357 70-5-1368	After upgrading, TELNET users from a terminal server who mistyped their password enough times could trigger the OpenVMS Intrusion Detection mechanism and cause all TELNET users from that terminal server to be locked out. You can solve the problem of inadvertent intrusion lockout with TELNET in either of the following ways: • Require clients to use RLOGIN instead of TELNET. • Loosen the intrusion detection policies on the system through appropriate tuning of the SYSGEN LGI* parameters. If neither solution is desireable, you can set the logical name TCPIP$TELNET_NO_REM_ID. Defining this logical name reverts TELNET to its original behavior of not setting any SYS$REM* logical. Note that the use of this logical name effectively bypasses the intrusion-detection mechanism for TELNET logins.

Table 4–4 SMTP Problems Corrected in this Release

CFS Number	PTR Number	Description
CFS.87060	70-5-1855	The SMTP symbiont failed and the log file contained the following error: `SYSTEM-F-NOPRIV, error trying to access CF control file` The error message was followed by ACCVIO errors. This occurred under a heavy SMTP load. The cause of the problem was corrected in the BIND resolver, which SMTP uses heavily.

Table 4–5 Management Command Interface Problems Corrected in this Release

CFS Number	PTR Number	Description
CFS.88243	70-5-1904	While attempting to create a new proxy, TCP/IP Services generated the following error message: `%SYSTEM-E-INSFMEM, insufficient dynamic memory` The problem was caused by the TCP/IP Services startup procedure, which did not set the proxy cache size. This problem has been corrected.

(continued on next page)

Table 4–5 (Cont.) Management Command Interface Problems Corrected in this Release

CFS Number	PTR Number	Description
CFS.81412 CFS.80586	70-5-1588 70-5-1560	In certain cases, the TCP/IP management command SHOW HOST failed with ACCVIO errors when retrieving the BIND database. There are two ways that a zone can be transferred: either in multiple DNS messages (each containing one resource record), or in one large DNS message that contains many resource records. The large DNS message may also span several messages, but it is differentiated because each message contains multiple resource records. The BIND resolver, which implements the SHOW HOST command, was not equipped for such a large amount of data arriving at one time. This problem has been corrected.
CFS.85063	70-5-1746	Entering the SHOW HOST command twice caused the management command interface to hang. This problem has been corrected.

5
Documentation Update

This chapter describes updates to the information in the TCP/IP Services for OpenVMS product documentation.

5.1 Management Guide Update

The following chapters are updated in these release notes:

- Chapter 5, Configuring and Managing BIND, is replaced by Appendix C in these release notes, for Alpha systems.
- Chapter 12, Configuring and Managing NTP, is replaced by Appendix B in these release notes.
- A new chapter, Configuring and Managing the IMAP Server for OpenVMS Mail, is provided in these release notes. See Appendix A.

In addition, the information in the *Compaq TCP/IP Services for OpenVMS Management* guide is updated as follows:

- Section 19.3.5 XDM_XSESSION.COM File, is misleading. The following changes will be made to this section:

  1. **Existing Text:**

     XDM's default operation is to create a Common Desktop Environment (CDE) using the commands from the SYS$SYSTEM:TCPIP$XDM_ XSESSION.COM file:

     Replacement Text:

     XDM's default operation after login is controlled by the SYS$SYSTEM:TCPIP$XDM_XSESSION.COM file. This file first parses its P1 display parameter *nodename*[:*server*[.*screen*]] and creates the DECwindows display using the following command:

     ```
     $ SET DISPLAY/CREATE/NODE="''nodename'"/TRANSPORT=tcpip -
     _$ /SERVER='server/SCREEN='screen
     ```

     The default operation is to then create a Common Desktop Environment (CDE) using:

  2. **Existing Text:**

     ```
     $ DEFINE DECW$DISPLAY "''p1'"
     $ DEFINE display      "''p1'"
     $ @CDE$PATH:XSESSION.COM
     ```

     Replacement Text:

     ```
     $ @CDE$PATH:XSESSION.COM
     ```

3. **Existing Text:**

   At present, CDE is available only on Alpha systems in Version 1.2-4 or later and not at all on VAX systems. If the CDE command procedure XSESSION.COM is not found on the system, XDM looks for the DECwindows Desktop Session Manager startup command procedure, DECW$STARTSM.COM to initiate the session using the commands:

   Replacement Text:

   At present, CDE is available only on Alpha systems in version 1.2-4 or later of DECwindows Motif, and not at all on VAX systems. If the CDE command procedure XSESSION.COM is not found on the system, XDM looks for the DECwindows Desktop Session Manager startup command procedure, DECW$STARTSM.COM, to initiate the session using the command:

4. **Existing Text:**

```
$ SET DISPLAY/CREATE/NODE=nodename/TRANSPORT=TCPIP

$ @SYS$MANAGER:DECW$STARTSM.COM
```

   Before executing either of these command procedures, XDM looks for an XDM_XSESSION.COM file in the user's SYS$LOGIN directory. If found, XDM executes the file. Users can create a DECterm by adding the following DCL commands to their XDM_XSESSION.COM file:

```
$ SET PROC/PRIV=SYSNAM

$ SET DISPLAY/CREATE/NODE=workstation_display/TRANSPORT=TCPIP -
_$ /EXECUTIVE_MODE

$ CREATE/TERMINAL/WAIT/WINDOW_ATTRIBUTES=(ICON=nodename, -
_$ TITLE=window_title)
```

   For a complete description of the CREATE and SET DISPLAY commands and their qualifiers, use the DCL command HELP at the OpenVMS system prompt.

   Replacement Text:

```
$ @SYS$MANAGER:DECW$STARTSM.COM
```

   Before executing either of these command procedures (but after performing the $SET DISPLAY), XDM looks for an XDM_ XSESSION.COM file in the user's SYS$LOGIN directory. If the file is found, XDM executes that file instead, passing it both the full display specification *nodename*[:*server*[.*screen*]]" as P1, and just the node name as P2.

   Users then have full control over exactly what type of session they prefer to start. For example, to start a DECterm, the following DCL commands are placed into their XDM_XSESSION.COM file:

```
$ CREATE/TERMINAL/WAIT/WINDOW_ATTRIBUTES=(ICON="''p2'",TITLE=window_title)
```

   For a complete description of the CREATE command and its qualifiers, use the DCL command HELP at the OpenVMS system prompt.

- Chapter 4, Routing, does not describe the change in the handling of identical routes between TCP/IP Services Version 5.0 and Version 5.1. In Version 5.0, routes were chosen by round-robin selection, but this feature was disabled by default. In Version 5.1, by default, they are chosen based on reference count and use count, thus achieving true load balancing among identical

routes. If more than one default route is set, all default routes are used in load balancing; if a route fails, then attempts to use that route also fail.

- The chapter on Remote commands (R commands) omits mention of the TCPIP$RCP_SEND_FIX_FORMAT_AS_ASCII logical name.

  For more information, see Section 3.2.

- Section 13.4.3, SNMP Options, fails to mention the fact that an SNMP community name can be added using either the SET CONFIGURATION SNMP command or the SNMP configuration file. The community name can consist of any 7-bit printable ASCII character except asterisk (*). This behavior is consistent with RFC 1157.

- In Chapter 22, the description of the LPD configuration logical TCPIP$LPD_KEEPALIVE is incorrect. The logical is set using the following command:

  ```
  $ DEFINE/SYSTEM TCPIP$LPD_KEEPALIVE 1
  ```

- In Chapter 15, Configuring and Managing FTP, the description of the FTP configuration logical TCPIP$FTP_KEEPALIVE is incorrect. To set this configuration parameter, enter the following command:

  ```
  $ DEFINE/SYSTEM TCPIP$FTP_KEEPALIVE 1
  ```

  The SET SERVICE command cannot be used to set this configuration parameter.

- The NFS Client chapter, Section 21.1.2.1 contains the following sentences, which are incorrect:

  ```
  (The client does not provide ADFs for files with the .TXT and .C
  extensions, because these are STREAM_LF.) The client maintains these
  ADFs on the server.
  ```

  These sentences will be removed from the manual.

- Also in the NFS Client chapter, the example of the MOUNT command in Section 21.4.1 shows an incorrect command line. The first line of the example should be replaced by the following:

  ```
  TCPIP> MOUNT DNFS1:[A] /HOST=BART /PATH="/DKA0/ENG"
  ```

- In Chapter 13, Table 13-3, the logical name SNMP_SUPPRESS_LOGGING_TIMESTAMP is incorrectly documented. The name of the logical is TCPIP$SNMP_SUPPRESS_LOGGING_TIMESTAMP.

  Also, in Table 13-6, the logical name TCPIP$SNMP_TRACE is incorrectly documented. The name of the logical is SNMP_TRACE.

5.2 User's Guide Update

The information in the *Compaq TCP/IP Services for OpenVMS User's Guide* is updated as follows:

- This release supports the use of double quotes ("*name*") in the OpenVMS personal name. If the SMTP mailer encounters double quotes in the personal name, it changes them to single quotes ('*name*').

5.3 Management Command Reference Update

The information in the *Compaq TCP/IP Services for OpenVMS Management Command Reference* manual is updated as follows:

- The SHOW NFS_SERVER command displays NFS server statistics and parameters. You can specify the following qualifiers with the SHOW NFS_SERVER command:
  - /CONTINUOUS=*seconds*

    Optional. Defaults: Static display; /CONTINUOUS=4.

    Provides a dynamic display, with optional screen-update interval.

    To terminate the display, press Ctrl/Y.
  - /RPC

    Optional.

    Displays only RPC-related performance counters and statistics.
  - /SERVER

    Optional.

    Displays NFS server-related performance counters and statistics.
  - /VERSION=*version*

    Optional. Default: Display Version 2 and Version 3

    Displays version-specific NFS server performance counters and statistics.

    Versions can be specified as follows:

/VERSION=V2	Specifies only Version 2.
/VERSION=V3	Specifies only Version 3.
/VERSION=(V2,V3)	Specifies both Version 2 and Version 3. This is the default if you omit the /VERSION qualifier.
- The SET NFS_SERVER commands allows you to specify NFS server configuration information. Contrary to the information in manual, the /THREADS qualifier is not a valid qualifier. To modify this value, use the `sysconfigdb` utility to add an entry to the SYSCONFIGTAB file, as described in Section 1.6.
- When you use the TCP/IP management command DISMOUNT/HOST, you must include the /ALL qualifier.
- If you specify the /MASK qualifier on the SET ROUTE command, you must also include the /NETWORK qualifier.
- Explanations of the use of the probe timer and the drop count in the description of the SET PROTOCOL command are corrected as follows:
  - The /PROBE_TIMER qualifier specifies, in seconds, the length of time that TCP/IP Services will wait for a response when establishing a new TCP connection, as well as the time between idle probes when the SO_KEEPALIVE option is set.
  - The /DROP_COUNT qualifier specifies the number of idle probes that can go unsatisfied before a TCP connection is closed.
- The description of the SET CONFIGURATION ENABLE [NO]SERVICE command is incorrect. The description should read as follows:

These commands modify service-related information in the permanent configuration database that enable services for startup:

- on each node (node-specific)
- on every node in a cluster (cluster-wide)

SET CONFIGURATION ENABLE SERVICE adds an entry for a service to the list of enabled services in the configuration database.

SET CONFIGURATION ENABLE NOSERVICE deletes an entry for a service from the list of enabled services in the configuration database.

The FORMAT section should read as follows:

```
SET CONFIGURATION ENABLE [NO]SERVICE service
  [/COMMON ]
  [ /[NO]CONFIRM ]
```

Note that the *service* parameter is required and must always be specified. There is no default for the *service* parameter.

The PARAMETERS section should include the following:

service

Required.

Specifies the service to add or delete from the configuration database.

The QUALIFIERS section should include:

/COMMON

Optional. Default: node-specific service without /COMMON

Modifies service-related information in the configuration database for the clusterwide enabling or disabling of services.

/CONFIRM
/NOCONFIRM

Optional. Default: /CONFIRM with wildcards; otherwise, /NOCONFIRM

Used only with SET CONFIGURATION ENABLE NOSERVICE to control whether a request is issued before each delete operation to confirm that the operation should be performed.

The /CONFIRM qualifier requests user confirmation when deleting service records.

Enter one of the following at the confirmation prompt:

- Y to delete the record
- N to retain the record

The /NOCONFIRM qualifier eliminates all user confirmation when deleting service records.

The examples and supporting descriptions should read as follows:

1. `TCPIP> SET CONFIGURATION ENABLE SERVICE TELNET`

   In the configuration database, enable the TELNET service for startup on this node.

2. `TCPIP> SET CONFIGURATION ENABLE SERVICE FTP /COMMON`

   In the configuration database, enable the FTP service for startup on every node in the cluster.

3. TCPIP> SET CONFIGURATION ENABLE NOSERVICE *

```
Enable service
  TELNET
Remove? [N]:Y
```

In the configuration database, disable any service enabled for startup on this node if confirmed by the user.

The online Help file for the TCP/IP management interface has been updated with the changes listed in this section. For online help, enter the following commands:

```
$ TCPIP
TCPIP> HELP
```

5.4 Sockets API and System Services Programming Update

The information in the *Compaq TCP/IP Services for OpenVMS Sockets API and System Services Programming* manual should be updated as follows:

- Table 2-2 describes the default setting for the TCPIP_KEEPIDLE option incorrectly. The default setting for this option is 7200 seconds (14400 half seconds). In addition, the manual fails to mention that, in order to use the options in Table 2-2, your program must use the TCP.H file.

5.5 Help Files Update

The TCP/IP Services Help files should be updated, as described in the following sections.

5.5.1 The netstat Help File

the following options should be added to the `netstat` Help file:

- `-s`

  This option displays statistics for each protocol.

- `-pudp`

  This option displays statistics for the UDP protocol.

- `-ptcp`

  This option displays statistics for the TCP protocol.

- `-picmp`

  This option displays statistics for the ICMP protocol.

- `-ip`

  This option displays statistics for the IP protocol.

- `-b`

  This option displays the contents of the Mobile IPv6 binding cache. When used with the `-s` option, it displays binding cache statistics.

- `-f` *address_family*

  This option limits reports to the specified address family. Specify one of the following:

  - `inet6` for the AF_INET6 family
  - `inet` for the AF_INET family

The routing table display is updated with the following new flags:

- C indicates a cloning route that was created by the ROUTE command.
- c indicates a cloned route.
- f indicates fragment to path MTU size is disabled on this route
- I indicates a route that contains valid link-layer information.
- L indicates a loopback route that was created by the kernel.
- m indicates a route that was created by a Mobile IPv6 binding update.
- P indicates a route that was created by the Path MTU discovery process.
- p indicates that Path MTU discovery is disabled on this route.

5.5.2 The whois Help File

The whois is available in this release, but no Help file is available. To define the whois command, enter the following command:

```
$ whois :== $SYS$SYSTEM:TCPIP$WHOIS.EXE
```

The whois utility looks up user, host, and organization names in the Network Information Center (NIC) database.

The syntax for the whois command is as follows:

```
$ whois [-h server]
```

The -h option allows you to specify a whois server other than the default (rs.internic.net).

The *server* name can specify any of the following:

- The name of the registered user.
- The name of a registered Internet host.
- The name of some other entity recognized by the whois server. By default, the whois command queries the host rs.internic.net.

The operands specified for the whois command are concatenated together (separated by spaces) and presented to the whois server. The default action, unless directed otherwise with a special name, is to do a very broad search, looking for matches to name in all types of records and most fields (such as name, nicknames, host name, and network address) in the database.

For example:

```
$ whois osf.org
Open Software Foundation (OSF-DOM)
11 Cambridge Center
Cambridge, MA  02142

Domain Name:  OSF.ORG
   .
   .
   .
```

5.6 Guide to IPv6 Update

The information in the *Compaq TCP/IP Services for OpenVMS Guide to IPv6* should be updated with the information in Appendix D.

A

Configuring and Managing the IMAP Server for OpenVMS Mail

The IMAP Server for OpenVMS Mail and the Simple Mail Transfer Protocol (SMTP) server software work together to provide reliable mail management in a client/server environment.

The IMAP Server allows users to access their OpenVMS Mail mailboxes by clients such as Microsoft Outlook so that they can view, move, copy and delete messages. The SMTP Server provides the extra functionality of allowing the clients to create and send e-mail messages.

After the IMAP Server is enabled on your system, you can modify the default characteristics by editing the configuration file (described in Section A.2.3).

This chapter reviews key IMAP concepts and describes:

- How to start up and shut down the IMAP Server (Section A.2.1)
- How to modify IMAP server characteristics (Section A.2.3)
- How to enable MIME mail using IMAP (Section A.3)

A.1 Key Concepts

IMAP stands for Internet Message Access Protocol. The IMAP Server allows users to access their OpenVMS Mail mailboxes by clients communicating with the IMAP4 protocol as defined in RFC 2060. The supported clients used to access e-mail are PC clients running Microsoft Outlook or Netscape Communicator.

The IMAP server is by default assigned port number 143, and all IMAP client connections are made to this port.

The following sections review the IMAP process and describe how the TCP/IP Services software implements IMAP. If you are not familiar with IMAP, refer to RFC 2060 or introductory IMAP documentation for more information.

A.1.1 IMAP Server Process

The IMAP Server is installed with SYSPRV, BYPASS, DETACH, SYSLCK, SYSNAM, NETMBX, and TMPMBX privileges. It runs in the TCPIP$IMAP account, which receives the correct quotas from the TCPIP$CONFIG procedure. The IMAP Server is invoked by the auxiliary server.

The IMAP Server uses security features provided in the protocol and in the OpenVMS operating system, as well as additional security measures. These methods provide a secure process that minimizes the possibility of inappropriate access to a user's mail file on the served system.

You can modify the IMAP Server default characteristics and implement new characteristics by defining the configuration options described in Section A.2.3.

A.1.2 How to Access Mail Messages from the IMAP Server

The only client configuration that is required is in the user's IMAP client, (see Section A.1.2.1). Additional optional configuration settings can be made in the OpenVMS Mail utility (see Section A.1.2.2).

A.1.2.1 IMAP Client Configuration

To access mail messages from the IMAP Server, you configure a user name and password into your client mail application. If an account is accessible without a password and a password is provided, the password is ignored.

In OpenVMS Mail, a user's mailbox file is, by default, named MAIL.MAI and is resident in the user's default OpenVMS directory. In this simple and typical case, the user name to be configured is the user's OpenVMS account name.

An OpenVMS Mail user is allowed to have many mail files; a special syntax for the user name is defined so that the user can specify the set of mail files to be opened. This syntax is the user's OpenVMS account name followed by a percent sign and then partial file specifications of the mail files, each separated by a percent sign. Note the following:

- The file extension of .MAI is not required.
- The default MAIL.MAI file is assumed, but if the user adds it to the list, it will not be displayed twice.

In the following example, user SMITH has three mail files in a mail directory. One is the user's default MAIL.MAI file, and the others are ACCOUNTS.MAI and PRIVATE.MAI. The user name would be configured in the IMAP client as:

```
SMITH%ACCOUNTS%PRIVATE
```

Your client system opens the TCP connection and attempts to access the server by entering the IMAP LOGIN command with the configured user name and password. On successful connection, the user's mail files are the top level of mailboxes; so, in the preceding example, the mailboxes displayed will be Mail, Accounts, and Private. Unsuccessful attempts are logged in the event log file (see Section A.2.2).

Once an OpenVMS Mail user has successfully connected to the IMAP Server, a file called tcpip$imap_mailbox.dat will have been created in the user's OpenVMS mail directory. This file is a text file and has a record of the mailbox files specified by the user. The file contains a newline separated list of partial file specifications (completed by each user's mail directory and the .MAI suffix)

As a result of this, the following possibilities exist:

- Next time the user connects to the IMAP Server, the user may optionally remove percent signs and mailbox names from the user name field since they will have been recorded permanently in tcpip$imap_mailbox.dat. It will not make any difference if the user opts to leave the mailbox names in the user name field.
- The user may add extra mailbox names in a later connection, and the file tcpip$imap_mailbox.dat will be updated to provide a consolidated list.
- The IMAP Client software may create and delete mail files in the user's directory. These actions may be at the user's behest, or performed automatically such as the creation of a "Deleted Items" mailbox.

- A System Manager may wish to provide every user on the system with a preloaded tree of mailboxes, and this can be achieved by copying to each user's mail directory a pre-formatted copy of tcpip$imap_mailbox.dat.

A.1.2.2 OpenVMS Mail Configuration

Table A–1 describes the how the IMAP Server is affected by the settings of the OpenVMS Mail SET options.

Table A–1 OpenVMS Mail SET options

Option	Description
AUTO_PURGE	IMAP clients have their own purge command, often called Compact or Compress. When a message is purged from a non-Wastebasket mailbox, it is put in the OpenVMS Mail Wastebasket. If the Wastebasket itself is purged, the messages are permanently deleted (equivalent to OpenVMS Mail PURGE), and disk space is then reclaimed as soon as the IMAP client disconnects. This occurs regardless of the AUTO_PURGE setting, which has no affect for an IMAP client. Any messages that are permanently deleted by an IMAP client will briefly appear in the user's Wastebasket but those messages will not be visible from the IMAP client and will be purged each time a client disconnects from that mail file.
CC_PROMPT	Not applicable.
COPY_SELF	Not applicable.
EDITOR	Not applicable.
FILE	Not used. Instead the user can configure all mail files to be available simultaneously by the setting of the user name in the IMAP client configuration, as described in Section A.1.2.1. Note that it is not possible to copy or move messages between different mail files.
FOLDER	Not applicable.
FORM	Not applicable.
FORWARD	This is not applicable to the IMAP Server, but setting a forwarding address results in messages being forwarded by OpenVMS Mail before being seen by the IMAP Server.
MAIL_DIRECTORY	This setting is respected. The directory, with an extension of .MAI, forms the partial file specification used to complete the file names of mail files using the values supplied as part of the user name in the IMAP client configuration. See Section A.1.2.1 for details.
PERSONAL_NAME	Not applicable.
QUEUE	Not applicable.
SIGNATURE_FILE	Not applicable.
WASTEBASKET_NAME	This setting is respected. This folder is always displayed by the IMAP client, even if it is currently empty.

A.1.3 How OpenVMS Mail Folder Names Map to IMAP Mailbox Names

OpenVMS Mail folders are presented to the IMAP client as IMAP mailboxes. All mailboxes are presented to the client in lowercase characters, beginning with an initial capital letter, and with capital letters following each space, at sign (@), opening parenthesis ("("), underscore (_), and hyphen (-).

The NEWMAIL folder requires special treatment . Because the IMAP protocol requires a top-level mailbox called Inbox, the NEWMAIL folder is mapped to Inbox. When the user opens the mailbox called Mail (which maps to file MAIL.MAI), the NEWMAIL folder is not listed so that the user is not confused by seeing the same folder listed twice.

OpenVMS Mail folder names are usually in all uppercase characters but can contain lowercase characters. Any lowercase characters are mapped to an underscore (_) followed by the character's uppercase equivalent. Underscores are mapped to double underscores (__), and dollar signs are mapped to double dollar signs ($$).

Table A–2 shows effects of folder-name mapping.

Table A–2 OpenVMS Mail Folder-Name Mapping

OpenVMS Mail Folder Name	IMAP Mailbox Name
HELLO	Hello
Hello	H_e_l_l_o
HELLO-ALL	Hello-All
HELLO_ALL	Hello__All
HELLO$ALL	Hello$$All

A.1.4 How the IMAP Server Handles Foreign Message Formats

The IMAP Server determines the correct format for common file types. It does this by checking the beginning of the file for a recognizable file header that matches a set contained in the configuration file TCPIP$IMAP_HOME:TCPIP$IMAP_MAGIC.TXT (analogous to the magic files found on UNIX systems). If a matching file header is found, the server can let the client know the MIME type and subtype of the file.

Though most common file formats are included, it is possible to add other formats if the structure of the file header is known.

Table A–3 describes the format of file-header recognition records.

Table A–3 IMAP File-Header Recognition

Field	Description
Type	The type of the data to be tested. Possible values are: • byte: A 1-byte value • short: A 2-byte value • long: A 4-byte value • string: A string of bytes • integer: An integer can be used indicating a number of bytes (30 or fewer) to allow for long matches. Examples in the configuration file are the two RIFF format files of WAVE and AVI. Optionally, types can be followed by an ampersand (&) and a numeric value (mask), expressed in hexadecimal, to specify that the value is to be AND'ed with the numeric value before any comparisons are done. Examples in the configuration file are the two RIFF format files of WAVE and AVI.
Test	The value to be compared with the value from the file. The rules for the value depend on the type: • Numeric type: The value is specified as octal or hexadecimal in the C programming language form where, for example, 013 is octal, and 0x13 is hexadecimal. • String type: The value is specified as a C programming language string with the usual escapes permitted (for example, `\n` to indicate new line.)
Type	The Content-Type type to use in the MIME version of this message.
Subtype	The Content-Type subtype to use in the MIME version of this message.
Format	Either "text" or "foreign", indicating what sort of OpenVMS Mail message will be tested for this match. A test with "foreign" in this field is performed on messages only if they are sent using the /FOREIGN qualifier. Note that this implicitly advertises whether base64 or quoted-printable encoding will be used for a given bodypart type; base64 is used only if the message was sent using the /FOREIGN qualifier.

A.1.5 Understanding IMAP Message Headers

Mail message headers sent by the IMAP Server must conform to the standard specified in RFC 822. Because many of the messages received on an OpenVMS system are not in the RFC 822, or Internet, format (for example, DECnet mail or mail from another message transport system), the IMAP Server builds a new set of headers for each message that is not RFC 822 format and is based on the OpenVMS message headers.

The following table describes the headers on mail messages that are forwarded by the IMAP Server.

IMAP Message Header	Obtained From
Date:	Arrival date of message. Changed to Internet format, which shows the day of the week, the date, the time, and the time zone offset from Greenwich Mean Time (GMT). An example of the format is Wed, 30 May 01 16:19:53 +0100.
From:	OpenVMS message From: field. Rebuilt to ensure RFC 822 compatibility. (See Section A.1.5.1.)
To:	OpenVMS Mail To: field. Rebuilt to ensure RFC 822 compatibility. (See Section A.1.5.1.)
CC:	OpenVMS Mail CC: field. Rebuilt to ensure RFC 822 compatibility. (See Section A.1.5.1.)
Subject:	OpenVMS Mail Subj: field. Accented characters are RFC 2047 encoded, but the change is not visible to users because IMAP clients reverse the encoding.
X-VMS-From:	OpenVMS Mail From: field. Not rebuilt.
X-IMAP4-Server:	Server host name and IMAP version information. Sent only if configuration option Send-ID-Headers is set to True.
X-IMAP4-ID:	Message UID. Sent only if configuration option Send-ID-Headers is set to True.

The IMAP Server sends these message headers to the IMAP client unless both of the following conditions are true:

- The configuration option Ignore-Mail11-Headers is set to True or is not defined (see Section A.2.3).
- The message text starts with SMTP headers.

A.1.5.1 How IMAP Rebuilds OpenVMS Mail Address Fields

It is important for the IMAP Server to rebuild the From: header, because this header can be used as a destination address if a reply is requested from the IMAP client. The same is true for To: and CC: headers if the user requests that a reply be sent to other listed recipients. Therefore, the IMAP server rebuilds these fields in compliance with RFC 822 before sending the header to the IMAP client.

The following table describes the different types of addresses that can appear in the OpenVMS Mail address fields.

Address Type	Address Format
SMTP	SMTP%"*legal-address*", where *legal-address* is an address that is compliant with RFC 822 and is commonly in the format *user@domain*.
DECnet	*node::username*
User name	*username*
DECnet address containing quotation marks	*node::"user@host"*
Cluster-forwarding SMTP address	*node*::SMTP%*"user@domain"*

A host name is local if one of the following conditions is true:

- The host name is the same as the substitute domain specified in the SMTP configuration.

- The host name is found in the TCPIP$SMTP_LOCAL_ALIASES.TXT file.

Some IMAP client systems are confused by the use of personal names when you attempt to reply to a mail message or when the name contains commas or other special characters. If you define the configuration option Personal-Name described in Section A.2.3, then before going live make sure you test the configuration carefully with your IMAP client systems to ensure that message replies work successfully.

The following sections describe how IMAP rebuilds an address field for each type of address.

A.1.5.1.1 SMTP Address The IMAP server uses the SMTP address within the quotation marks to rebuild the address field of an SMTP address. For example, message header `From: SMTP%"john.smith@federation.gov"` becomes:

```
From: john.smith@federation.gov
```

SMTP hides nested quotation marks by changing them to cent sign (¢) characters before passing them to OpenVMS Mail and then changing them back after a reply. The IMAP server removes any cent signs that designate double quotation marks. For example, the following message header:

```
From: SMTP%"¢ABCMTS::MRGATE::\¢ABCDEF::VIVALDI \¢¢@xyz.org"
```

Becomes:

```
From: "ABCMTS::MRGATE::\"ABCDEF::VIVALDI\""@xyz.org"
```

A.1.5.1.2 DECnet Address The value assigned to the configuration option Decnet-Rewrite defines how the IMAP Server rebuilds a DECnet address. The following list describes the possible values:

- GENERIC

  The entire address is changed to the SMTP format. For example, from host `widgets.xyzcorp.com`, the message header `From: ORDERS::J_SMITH` becomes:

  ```
  From: "ORDERS::J_SMITH"@widgets.xyzcorp.com
  ```

  In this example, instead of `widgets.xyzcorp.com`, the value of configuration option Gateway-Node (described in Section A.2.3) is used if it is defined; if not, the value of the SMTP substitute domain is used. Only if both of these options are undefined is the host name `widgets.xyzcorp.com` used.

- NONE

  The `From:` line is sent to the IMAP client unmodified. For example:

  ```
  From: ORDERS::J_SMITH
  ```

  You cannot reply to this type of message from an IMAP client because the SMTP server does not accept an address in this form.

- TRANSFORM

  The IMAP Server attempts to translate the DECnet node name to a TCP/IP host name. If the name can be translated, the IMAP server checks whether the translated host name is local. If so, the `From:` header becomes an address in the format *user@substitute-domain*. If not, the `From:` header becomes an address in the format *user@hostname*. Note that the IMAP and SMTP servers call the same routine to determine whether a host name is local.

The following examples show some ways the IMAP Server translates DECnet node names to TCP/IP node names. In these examples:

- The local host name is orders.acme.widgets.com
- ORDERS translates the name to "orders.acme.widgets.com"
  * The message header From: ORDERS::J_SMITH becomes:

    ```
    From: j_smith@orders.acme.widgets.com
    ```

  * For a substitute domain of acme.widgets.com, the message header From: ORDERS::J_SMITH becomes:

    ```
    From: j_smith@acme.widgets.com
    ```

  * If HOST12 translates to host12.acme.widgets.com, which is not local on host name orders.acme.widgets.com, the message header From: HOST12::J_SMITH becomes:

    ```
    From: j_smith@host12.acme.widgets.com
    ```

  * If HOST13 does not translate and host orders.acme.widgets.com has no substitute domain defined, the message header From: HOST13::J_SMITH becomes:

    ```
    From: "HOST13::J_SMITH"@orders.acme.widgets.com
    ```

    In this example, if the configuration option Gateway-Node is defined, then its value is used instead of orders.acme.widgets.com.

A.1.5.1.3 User Name-Only Address If the address under consideration is a recipient address, and the From: address is a DECnet address, then the recipient address is prefixed with the same routing information as that of the From: address. Then it is processed as if it were a DECnet address, as shown in Section A.1.5.1.2.

Otherwise the IMAP Server appends the at sign (@) to the user name, and then appends one of the following, in order of preference:

- The value of configuration option Gateway-Node, if defined
- An SMTP substitute domain, if defined
- The local host name

For example, with an SMTP substitute domain defined as acme.widgets.com, the message header From: Smith becomes:

```
From: smith@acme.widgets.com
```

A.1.5.1.4 DECnet Address That Contains Quotation Marks The values assigned to the configuration option Quoted-Decnet-Rewrite define how the IMAP Server rebuilds a DECnet address that contains quotation marks. The following list describes the possible values:

- GENERIC

  The address is changed to the SMTP format. For example, on host widgets.xyzcorp.com, the message header From: ORDERS::"j_smith@acme.com" becomes:

  ```
  From: "ORDERS::\"j_smith@acme.com\""@widgets.xyzcorp.com
  ```

- NONE

The From: line is passed to the IMAP client without being modified. For example:

```
From: ORDERS::"j_smith@acme.com"
```

You cannot reply to this type of mail message because the SMTP server does not accept an address of this form.

- TRANSFORM

  The IMAP Server uses the text inside the quotation marks. For example, the message header From: ORDERS::"j_smith@acme.com" becomes:

  ```
  From: j_smith@acme.com
  ```

A.1.5.1.5 Cluster-Forwarding SMTP Address With a cluster-forwarding SMTP address, the IMAP server uses the SMTP address within the quotation marks. For example, the message header From: ABCDEF::SMTP%"james.smith@federation.gov" becomes:

```
From: james.smith@federation.gov
```

A.1.5.1.6 All Other Addresses For all other address formats, the IMAP server changes the entire address to the SMTP format:

- Quotation marks in the address are prefixed with the backslash (\) escape character.
- The entire address is placed within quotation marks.
- An at sign (@) is appended.
- The value of configuration option Gateway-Node, if defined, is appended; if not, the value of the SMTP substitute domain is appended. If both are undefined, then the name of the local host is appended.

For example, if the substitute domain is xyz.org, the message header From: ABCMTS::MRGATE::"ORDERS::SPECIAL" becomes:

```
From: "ABCMTS::MRGATE::\"ORDERS::SPECIAL\""@xyz.org
```

If the configuration option Ignore-Mail11-Headers is set to True and the address is an SMTP address, the rebuilt From: field is not displayed to the user. In this case, the IMAP Server sends the actual headers from the body of the mail as the mail headers.

A.1.6 Uploaded Messages

A user can copy mail messages stored on the local client system to OpenVMS Mail. This action is termed **uploading** and involves the creation of a new OpenVMS Mail message.

When a message is uploaded, OpenVMS Mail treats it as a new mail message, and a "New Mail" broadcast message is issued. The user will see this message if the user also has an OpenVMS VT session open with receipt of broadcast messages enabled.

When a message is uploaded, the entire message is copied along with the header information described in Table A–4. Note that the additional header information is visible only if the user reads it with MAIL or if the configuration option Ignore-Mail11-Headers is set to False.

Table A–4 describes the typical headers in an uploaded message.

Table A–4 Header Information in Uploaded Messages

Header	Value
`Body:`	The entire SMTP message, including headers.
`From:`	The underscore character (_), followed by the name of the user who is uploading the message
`To:`	The underscore character (_), followed by the name of the user who is uploading the message
`Subj:`	The subject of the uploaded message.

A.2 IMAP Server Control

The system manager controls the management functions of the IMAP Server. These functions include:

- Starting and stopping the server
- Viewing event logs for each server
- Modifying options that control server behavior
- Tuning the server

The following sections describe these management functions.

A.2.1 Starting Up and Shutting Down the Server

The IMAP Server process starts automatically if you specified automatic startup during the configuration procedure (TCPIP$CONFIG.COM).

The IMAP Server can be shut down and started independently of TCP/IP Services. This is useful if you change configuration options that require the service to be restarted.

The following files are provided:

- SYS$STARTUP:TCPIP$IMAP_STARTUP.COM allows you to start the IMAP Server.
- SYS$STARTUP:TCPIP$IMAP_SHUTDOWN.COM allows you to shut down the IMAP Server.

To preserve site-specific parameter settings and commands, create the following files:

- SYS$STARTUP:TCPIP$IMAP_SYSTARTUP.COM — to be used as a repository for site-specific definitions and parameters to be invoked when the IMAP server is started.
- SYS$STARTUP:TCPIP$IMAP_SYSHUTDOWN.COM — to be used as a repository for site-specific definitions and parameters to be invoked when the IMAP Server is shut down.

Note that these files are not overwritten when you reinstall TCP/IP Services.

A.2.2 Viewing Server Event Log Files

The IMAP Server records start and stop server events in an event log file. Other events, such as failed user authentication events, are also recorded in this log file. The file is called TCPIP$IMAP_HOME:TCPIP$IMAP_EVENT$*node*.LOG, where *node* is the name of the node on which the server is running.

A.2.3 Modifying IMAP Server Characteristics

To modify the default IMAP Server settings and to configure additional characteristics, edit the configuration file TCPIP$IMAP_HOME:TCPIP$IMAP.CONF. If you modify the IMAP Server configuration file, restart the IMAP Server to make the changes take effect.

You can modify the following IMAP Server characteristics:

- Server port number
- Mail header options
- Number of live connections per server process

The format of each line in the configuration file is:

option:*value*

where *option* is the setting name and *value* is the value given to the setting. For example:

```
Server-Port:143
```

Comment lines may be added to the configuration file, and these lines are defined to be lines that start with the '#' character.

Table A–5 describes the IMAP option names, default settings, and characteristics that you can modify.

Table A–5 IMAP Configuration Options

Option Name	Description
Server-Port	TCP/IP port number for connection between IMAP clients and the IMAP Server. The default value is 143.
Ignore-Mail11-Headers	If set to True, the default, the IMAP Server ignores the OpenVMS message headers when mail is sent via SMTP, which contains an SMTP address in the `From:` field. For information about how IMAP forms message headers, see Section A.1.5.
Send-ID-Headers	If set to True, the IMAP Server sends `X-IMAP4-Server` and `X-IMAP4-ID` headers for each mail message. If not defined or if set to False (the default), the ID headers are not sent for any mail from an SMTP address. For information about how IMAP handles message headers, see Section A.1.5.

(continued on next page)

Table A–5 (Cont.) IMAP Configuration Options

Option Name	Description
Decnet-Rewrite	Determines how the IMAP Server rebuilds a simple DECnet address (of the form *node*::*user*) when it sends the mail to the IMAP client The value of this option may be one of the following: • GENERIC Simple DECnet addresses are changed to the SMTP address format. • NONE Simple DECnet addresses are sent unmodified to the IMAP client. • TRANSFORM (default) The IMAP server attempts to transform the DECnet address into an SMTP address by translating the DECnet node name to a TCP/IP host name. For more information about how IMAP rebuilds the message headers, see Section A.1.5.1.2.
Quoted-Decnet-Rewrite	Determines how the IMAP Server rebuilds a DECnet address that contains quotation marks (of the form *node*::*"user@host"*) in the OpenVMS Mail `From:` field when it sends the message to the IMAP client. The value of this option may be one of the following: • GENERIC DECnet addresses that contain quotation marks are changed to the SMTP address format. • NONE DECnet addresses that contain quotation marks are sent unmodified to the IMAP client. • TRANSFORM (default) The IMAP server uses the text within the quotation marks in the `From:` field it sends to the IMAP server. For more information about how IMAP rebuilds the message headers, see Section A.1.5.1.4.
Personal-Name	If defined, the IMAP server provides the IMAP clients with the message header `From:` fields that include the sender's personal name, if one appeared in the sender's `From:` field.
Gateway-Node	If defined, the local node or cluster name is superseded by the value of this configuration option, when supplying a route from SMTP into DECnet as part of an address. The Gateway-Node value should be an Internet address of a TCP/IP Services SMTP server node. For example, suppose a Decnet node name of ORDERS cannot be mapped, and the address is ORDERS::J_SMITH and Gateway-Node is defined to be widgets.xyzcorp.com, then the resulting address will be "ORDERS::J_ SMITH"@widgets.xyzcorp.com.

(continued on next page)

Table A–5 (Cont.) IMAP Configuration Options

Option Name	Description
Max-Connections	Each time the number of live connections to the server reaches the Max-Connections parameter (default = 25), a new process is started. The old server does not accept new connections and will shut down as soon as all existing connections are closed by the client or after 20 minutes, whichever comes first. Service may be interrupted for up to 5 seconds while one process takes over from the other. You should avoid changing the value of this option unless instructed by Compaq support personnel. Too low a setting will result in unnecessary delays, and too high a setting will result in too many connections contesting per-process-limited OpenVMS Mail resources.
Message-Cap	The IMAP Server has a configurable limit on the number of messages displayed in a folder, defined by the Message-Cap parameter. If this parameter is not defined, or is set to the default of 0, then no limit is applied. If a user tries to list a folder containing more messages than the limit, then only the first n messages will be displayed, where n is the limit. In addition the user will be informed with the message "Only the first n messages in each folder will be displayed. Delete or move messages to another folder to display more."
Server-Trace	The default for this setting is False. If set to True a trace file will be created as TCPIP$IMAP_HOME:TCPIP$IMAP_node_yyyymmddhhmmssTRACE.LOG where "node" is the node name where the IMAP Server is running and "yyyymmddhhmmss" is the time that the trace log is created. All communication between IMAP clients and the IMAP Server will be logged, with the exception of passwords. Since a large amount of data will be recorded on a busy system, it is recommended that tracing is only turned on for short periods. To turn tracing on or off, it is necessary to restart the IMAP Server.

(continued on next page)

Table A–5 (Cont.) IMAP Configuration Options

Option Name	Description
Trace-Synch	This setting only applies when the setting Server-Trace is set to True. Trace-Synch governs the frequency with which the trace log is flushed. Trace-Synch is a non-negative integer value which specifies the number of trace log writes between each full flush of the trace log output to disk. Flushing the log more frequently lowers the number of log writes that could be lost at the end of the log in the event of a server process crash but it also means slower performance. Conversely less frequent flushing means better performance but more lines possibly lost at the end of the log on a server crash. A value of 0, which is the default, means not to flush to disk until the server process exits, though OpenVMS Record Management Services (RMS) will flush to disk periodically anyway. This is the highest performance option but in the event of a server crash many lines of trace log information may be lost. If you want the server to flush on each log write set Trace-Synch to 1. This is the slowest performer but safest regarding potential loss of trace log data in the event of a server crash. Benchmark testing has proven that a value of 100 strikes a good balance between performance and data loss.

A.2.4 Tuning the Server

This section is intended for system managers who want to know more detailed information about the IMAP server.

A.2.4.1 Tuning Issues

The only tuning issue pertains to the server's use of Virtual Memory (VM). The IMAP server can use large amounts of VM and may require some tuning to get dependable performance.

The exhaustion of virtual memory can be manifested in different ways including the server process crashing with error messages that could appear to be caused by different problems such as access violations, ROPRAND as well as INSVIRMEM errors. Some crashes produce PTHREAD_DUMP.LOG files in TCPIP$IMAP_HOME.

Although the process crashes may appear to be from different causes, memory exhaustion can be confirmed by examining the job termination information at the tail of the TCPIP$IMAP_RUN.LOG. If the "Peak virtual size" value is at or above the TCPIP$IMAP account's PGFLQUO value then the process probably terminated due to insufficient virtual memory.

The IMAP server process will sometimes hang rather than exit when it consumes all of its dynamic memory. Use the following commands to examine the PAGFILCNT of the IMAP server process. If the value is at or near zero then the hang is caused by insufficient virtual memory.

```
$!
$! This shows the PID(s) of the IMAP process(es)...
$ SHOW SYSTEM/PROCESS=*IMAP*
$!
$! To show the process's PAGFILCNT do
$ WRITE SYS$OUTPUT "''F$GETJPI("insert-pid-here","PAGFILCNT")'"
```

A.2.4.2 Tuning Options

The use of virtual memory by the IMAP Server can be controlled by one or both of the following actions: increasing the PGFLQUO of the server account or reducing the memory requirements of the process.

These two options are described in more detail in the next sections.

A.2.4.2.1 Give more dynamic memory to an IMAP server process If sufficient memory is available, increase the PGFLQUO of the TCPIP$IMAP account (adjusting any SYSGEN parameters that limit PGFLQUO). Take into account that multiple server processes may need to run concurrently on a heavily loaded system when assigning a high PGFLQUO.

The amount of memory an IMAP server can use at peak is the sum of the following elements:

- Memory required at start-up: 10000 blocks

- Memory per connection: 1500 blocks. The number of connections per Server is limited by the Max-Connections configuration option, default 25. Thus in a default configuration the amount required is 37500 blocks.

- Memory per message: If a user lists a large folder, then the Server requires an extra 3 blocks per message. If you estimate that your users are opening, at peak, folders with an average of 1000 messages, then the amount of memory required will be:

  Max-Connections * 3 * 1000

  which is 75000 blocks in a default configuration. See also the next section where use of the configuration option Message-Cap is described, in that it can be used to limit the maximum number of messages in a folder that can be viewed. If Message-Cap is defined then the formula will be:

  Max-Connections * 3 * Message-Cap

A.2.4.2.2 Reduce IMAP server demand for memory There are two IMAP configuration options that can be used to reduce each IMAP server process's consumption of dynamic memory. They are Max-Connections and Message-Cap.

Because Max-Connections limits the number of connections that can be served simultaneously it will implicitly limit the quantity of VM consumed by each IMAP server process. Note that while reducing Max-Connections reduces the dynamic memory consumption of each IMAP server process it will result in more IMAP server processes; there will be more processes each using less dynamic memory.

Message-Cap limits the number of messages passed back to the client when a folder is selected.

A.3 Enabling MIME Mail

The MIME (Multipurpose Internet Mail Extensions) specification provides a set of additional headers you can use so that users can send mail messages composed of more than simple ASCII text. MIME is an enhancement to RFC 822.

For MIME mail to be decoded correctly, follow these guidelines:

- Configure the SMTP server with the /OPTION=TOP_HEADERS qualifier, because the first lines of mail text after the four OpenVMS message header lines and the initial separating line must be the MIME headers.
- Configure the IMAP Server with the option Ignore-Mail11-Headers set to True, or leave this option undefined, since True is the default value. Otherwise, MIME headers are not parsed as message headers.
- The OpenVMS message `From:` field must be recognized as an SMTP address. Otherwise, the IMAP server sends the headers it creates from OpenVMS message headers as the headers of the mail message. For information about IMAP message headers, see Section A.1.5.

  Define the logical name TCPIP$SMTP_JACKET_LOCAL to 1 for all SMTP cluster systems. This ensures that the mail is delivered if the domain in the `From:` or `To:` field appears local. For example:

  ```
  $ DEFINE/SYSTEM TCPIP$SMTP_JACKET_LOCAL 1
  ```

If MIME mail does not decode, check the mail headers on the client system. If you see multiple blocks of headers and the MIME version header is not in the first block, confirm that you have followed these guidelines. Note that the headers of messages forwarded over OpenVMS Mail are mapped only if there is no cover note (that is, if the headers of a forwarded message are at the top of the message immediately following the headers of the forwarding message).

B

Configuring and Managing NTP

The Network Time Protocol (NTP) synchronizes time and coordinates time distribution throughout a TCP/IP network. NTP provides accurate and dependable timekeeping for hosts on TCP/IP networks. TCP/IP Services NTP software is an implementation of the NTP Version 4 specification and maintains compatibility with NTP Versions 1, 2, and 3.

Time synchronization is important in client/server computing. For example, systems that share common databases require coordinated transaction processing and timestamping of instrumental data.

NTP provides synchronization that is traceable to clocks of high absolute accuracy and avoids synchronization to clocks that keep incorrect time.

This chapter reviews key concepts and describes:

- How to start up and shut down NTP (Section B.2)
- How to configure the NTP host (Section B.3)
- How to configure the host as a backup time server (Section B.4)
- NTP event logging (Section B.5)
- How to configure NTP authentication (Section B.6)
- How to use NTP utilities (Section B.7)
- How to solve NTP problems (Section B.8)

B.1 Key Concepts

Synchronized timekeeping means that hosts with accurate system timestamps send time quotes to each other. Hosts that run NTP can be either time servers or clients, although they are often both servers and clients.

NTP does not attempt to synchronize clocks to each other. Rather, each server attempts to synchronize to Coordinated Universal Time (UTC) using the best available source and the best available transmission paths to that source. NTP expects that the time being distributed from the root of the synchronization subnet will be derived from some external source of UTC (for example, a radio clock).

If your network is isolated and you cannot access other NTP servers on the internet, you can designate one of your nodes as the reference clock to which all other hosts will synchronize.

B.1.1 Time Distributed Through a Hierarchy of Servers

In the NTP environment, time is distributed through a hierarchy of NTP time servers. Each server adopts a stratum that indicates how far away it is operating from an external source of UTC. NTP times are an offset of UTC. Stratum 1 servers have access to an external time source, usually a radio clock. A stratum 2 server is one that is currently obtaining time from a stratum 1 server; a stratum 3 server gets its time from a stratum 2 server; and so on. To avoid long-lived synchronization loops, the number of strata is limited to 15.

Stratum 2 (and higher) hosts might be company or campus servers that obtain time from some number of primary servers and provide time to many local clients. In general:

- Lower-strata hosts act as time servers.
- Higher-strata hosts are clients that adjust their time clocks according to the servers.

Internet time servers are usually stratum 1 servers. Other hosts connected to an internet time server have stratum numbers of 2 or higher and may act as time servers for other hosts on the network. Clients usually choose one of the lowest accessible stratum servers from which to synchronize.

B.1.2 How Hosts Negotiate Synchronization

The identifying stratum number of each host is encoded within UDP datagrams. Peers communicate by exchanging these timestamped UDP datagrams. NTP uses these exchanges to construct a list of possible synchronization sources, then sorts them according to stratum and synchronization distance. Peers are accepted or rejected, leaving only the most accurate and precise sources.

NTP evaluates any new peer to determine whether it qualifies as a new (more suitable) synchronization source.

NTP rejects the peer under the following conditions:

- The peer is not synchronized.
- The stratum is higher than the current source's stratum.
- The peer is synchronized to the local node.

NTP accepts the peer under the following conditions:

- There is no current time source.
- The current source is unreachable.
- The current source is not synchronized
- The new source's stratum is lower than the current source.
- The new source's stratum is the same as the current source, but its distance is closer to the synchronization source by more than 50 percent.

B.1.3 How the OpenVMS System Maintains the System Clock

The OpenVMS system clock is maintained as a software timer with a resolution of 100 nanoseconds, updated at 10-millisecond intervals. A clock update is triggered when a register, loaded with a predefined value, has decremented to zero. Upon reaching zero, an interrupt is triggered that reloads the register, thus repeating the process.

The smaller the value loaded into this register, the more quickly the register reaches zero and triggers an update. Consequently, the clock runs more quickly. A larger value means more time between updates; therefore, the clock runs more slowly. A **clock tick** is the amount of time between clock updates.

B.1.4 How NTP Makes Adjustments to System Time

Once NTP has selected a suitable synchronization source, NTP compares the source's time with that of the local clock. If NTP determines that the local clock is running ahead of or behind the synchronization source, NTP uses a general drift mechanism to slow down or speed up the clock as needed. NTP accomplishes this by issuing a series of new clock ticks. For example, if NTP detects that the local clock is drifting ahead by +0.1884338 second, it issues a series of new ticks to reduce the difference between the synchronization source and the local clock.

If the local system time is not reasonably correct, NTP does not set the local clock. For example, if the new time is more than 1000 seconds off in either direction, NTP does not set the clock. In this case, NTP logs the error and shuts down.

NTP maintains a record of the resets it makes along with informational messages in the NTP log file, TCPIP$NTP_RUN.LOG. For details about event logging and for help interpreting an NTP log file, see Section B.5.

B.1.5 Configuring the Local Host

The system manager of the local host, determines which network hosts to use for synchronization and populates an NTP configuration file with a list of the participating hosts.

NTP hosts can be configured in any of the following modes:

- Client/server mode

  This mode indicates that the local host wants to obtain time from the remote server *and is willing* to supply time to the remote server. This mode is appropriate in configurations involving a number of redundant time servers interconnected through diverse network paths. Internet time servers generally use this mode.

  Indicate this mode with a `peer` statement in the configuration file, as shown in the following example:

  ```
  peer 18.72.0.3
  ```

- Client mode

  This mode indicates that the local host wants to obtain time from the remote server *but it is not willing* to provide time to the remote server. Client mode is appropriate for file server and workstation clients that do not provide synchronization to other local clients. A host with higher stratum generally uses this mode.

Indicate client mode with the server statement in the configuration file, as shown in the following example:

```
server 18.72.0.3
```

- Broadcast mode

  This mode indicates that the local server will send periodic broadcast messages to a client population at the broadcast/multicast address specified. This specification normally applies to the local server operating as a sender.

  Indicate this mode with a broadcast statement in the configuration file, as shown in the following example:

  ```
  broadcast 18.72.0.255
  ```

- Multicast mode

  A multicast client is configured using the broadcast statement, but with a multicast group (class D) address instead of a local subnet broadcast address. However, there is a subtle difference between broadcasting and multicasting. Broadcasting is specific to each interface and local subnet address. If more than one interface is attached to a machine, a separate broadcast statement applies to each one.

  IP multicasting is a different paradigm. A multicast message has the same format as a broadcast message and is configured with the same broadcast statement, but with a multicast group address instead of a local subnet address. By design, multicast messages travel from the sender via a shortest-path or shared tree to the receivers, which might require these messages to emit from one or all interfaces but to carry a common source address. However, it is possible to configure multiple multicast group addresses using multiple broadcast statements. Other than these differences, multicast messages are processed just like broadcast messages. Note that the calibration feature in broadcast mode is extremely important, since IP multicast messages can travel far different paths through the IP routing fabric than can ordinary IP unicast messages.

  The Internet Assigned Number Association (IANA) has assigned multicast group address 224.0.1.1 to NTP, but you should use this address only where the multicast span can be reliably constrained to protect neighbor networks. In general, you should use group addresses that have been given out by your administrator, as described in RFC 2365, or GLOP group addresses, as described in RFC 2770.

- Manycast mode

  Manycasting is an automatic discovery and configuration paradigm new to NTP Version 4. It is intended as a means for a multicast client to survey the nearby network neighborhood for cooperating manycast servers, to validate them using cryptographic means, and to evaluate their time values with respect to other servers that might be in the vicinity. The intended result is that each manycast client mobilizes client associations with the best three of the available manycast servers and automatically reconfigures to sustain this number of servers if one or more fail.

  A persistent manycast client association is configured using the server statement, but with a multicast (class D) group address instead of an ordinary IP (class A, B, C) address. There can be as many manycast client associations as different group addressses.

Manycast servers configured with the `manycastserver` statement listen on the specified group address for manycast client messages. Note the distinction between a manycast client, which is configured with a `server` statement, and a manycast server, which is configured with a `manycastserver` statement.

If a manycast server is in range of the current time-to-live and is synchronized to a valid source and operating at a stratum level equal to or lower than the manycast client, the server replies to the manycast client message with an ordinary server-mode message.

The manycast client that receives this message mobilizes an ephemeral client association as in ordinary client/server mode, according to the matching manycast client template. Then the client polls the server at its unicast address in burst mode in order to set the host clock reliably and to validate the source. The client runs the NTP intersection and clustering algorithms, which discard all but the best three associations. The surviving associations then continue in ordinary client/server mode.

- Burst mode

  Two burst modes can be enabled in client/server mode using the `server` statement and the `iburst` and `burst` keywords. In either mode, a single poll initiates a burst of eight client messages at intervals randomized over a range of 1 to 4 seconds. However, the interval between the first and second messages is increased to about 16 seconds in order for a dialup modem to complete a call, if necessary.

  Received server messages update the NTP Version 4 clock filter, which selects the best (most accurate) time values. When the last client message in the burst is sent, the next received server message updates the system variables and sets the system clock in the usual manner, as if only a single client/server cycle had occurred. The result is not only a rapid and reliable setting of the system clock, but also a considerable reduction in network jitter.

  The `iburst` keyword can be configured when it is important to set the clock quickly, such as when an association is either first mobilized or first becomes reachable, or when the network attachment requires an initial calling or training procedure. The burst is initiated only when the server first becomes reachable and results in good accuracy with intermittent connections typical of PPP and ISDN services. Outlyers caused by initial dialup delays and other factors are avoided, and the client sets the clock within 30 seconds after the first message.

  The `burst` keyword can be configured in cases of excessive network jitter or when the network attachment requires an initial calling or training procedure. The burst is initiated at each poll interval when the server is reachable. The burst does produce additional network overhead and can cause trouble if used indiscriminately. It should be used only if the poll interval is expected to settle to values equal to or greater than 1024 seconds.

B.2 NTP Service Startup and Shutdown

The NTP service can be shut down and started independently of TCP/IP Services. The following files are provided:

- SYS$STARTUP:TCPIP$NTP_STARTUP.COM allows you to start the NTP service.

- SYS$STARTUP:TCPIP$NTP_SHUTDOWN.COM allows you to shut down the NTP service.

To preserve site-specific parameter settings and commands, create the following files. These files are not overwritten when you reinstall TCP/IP Services:

- SYS$STARTUP:TCPIP$NTP_SYSTARTUP.COM can be used as a repository for site-specific definitions and parameters to be invoked when the NTP service is started.
- SYS$STARTUP:TCPIP$NTP_SYSHUTDOWN.COM can be used as a repository for site-specific definitions and parameters to be invoked when the NTP service is shut down.

B.3 Configuring Your NTP Host

The NTP configuration file TCPIP$NTP.CONF contains a list of hosts your system will use for time synchronization. Before configuring your host, you must do the following:

1. Select time sources.
2. Obtain the IP addresses or host names of the time sources.
3. Obtain the version number of NTP that the hosts are running.

To ensure reliable synchronization, select multiple time sources that you are certain provide accurate time and that are synchronized to an Internet time server.

To minimize common points of failure, avoid synchronizing the following:

- The local host to another peer at the same stratum, unless the latter is receiving time from a lower stratum source to which the local host cannot connect.
- More than one host in a particular administrative domain to the same time server outside that domain.

To simplify configuration file maintenance, avoid configuring peer associations with higher stratum servers.

B.3.1 Creating the Configuration File

To create a configuration file for your local host, edit a copy of the file TCPIP$NTP.TEMPLATE (located in SYS$SPECIFIC:[TCPIP$NTP]) to add the names of participating hosts, then save the file as SYS$SPECIFIC:[TCPIP$NTP]TCPIP$NTP.CONF. This file is not overwritten when you install subsequent versions of TCP/IP Services.

Note

If a UCX version of NTP is configured on your system, your TCPIP$NTP.CONF file is created automatically and is populated with entries from the file UCX$NTP.CONF when you run the TCPIP$CONFIG procedure.

B.3.2 Configuration Statements and Options

In the following configuration statements, the various modes are determined by the statement keyword and the type of the required IP address. Addresses are classsed by type as (s) a remote server or peer (IP class A, B, and C), (b) the broadcast address of a local interface, (m) a multicast address (IP class D), or (r) a reference clock address (127.127.x.x).

NTP configuration statements are formatted as follows:

- peer *address* [key *ID*] [version *number*] [prefer] [minpoll *interval*] [maxpoll *interval*]

  server *address* [key *ID*] [version *number*] [prefer] [burst] [iburst] [minpoll *interval*] [maxpoll *interval*]

  broadcast *address* [key *ID*] [version *number*] [minpoll *interval*] [ttl *nn*]

  manycastclient *address* [key *ID*] [version *number*] [[minpoll *interval*] [maxpoll *interval*] [ttl *nn*]

  These four statements specify the time server name or address to be used and the mode in which to operate. The *address* can be either a DNS name or an IP address in dotted-quad notation.

  - peer — For type s addresses only, this statement mobilizes a persistent symmetric-active mode association with the specified remote peer. This statement should not be used for type b, type m, or type r addresses.
  - server — For type s and type r addresses only, this statement mobilizes a persistent client mode association with the specified remote server or local reference clock. This statement should not be used for type b or type m addresses.
  - broadcast — For type b and type m addresses only, this statement mobilizes a persisent broadcast mode association. Multiple statements can be used to specify multiple local broadcast interfaces (subnets) and/or multiple multicast groups. Note that local broadcast messages go only to the interface associated with the subnet specified, but multicast messages go to all interfaces.
  - manycastclient — For type m addresses only, this statement mobilizes a manycast client mode association for the multicast address specified. In this case, a specific address must be supplied that matches the address used on the manycastserver statement for the designated manycast servers.

    The manycastclient statement specifies that the local server is to operate in client mode with the remote servers that are discovered as the result of broadcast/multicast messages. The client broadcasts a request message to the group address associated with the specified address and specifically enabled servers respond to these messages. The client selects the servers providing the best time and continues as with the server statement. The remaining servers are discarded as if never heard.

  The following table describes the options to the previous statements:

Option	Description
key *ID*	For all packets sent to the address, includes authentication fields encrypted using the specified key identifier, an unsigned 32-bit integer. The default is no encryption.
version *number*	Specifies the version number to be used for outgoing NTP packets. Versions 1, 2, 3, and 4 are the choices. The default is 4.
prefer	Marks the server as preferred. This host will be chosen for synchronization among a set of correctly operating hosts.
burst	When the server is reachable and at each poll interval, send a burst of eight packets instead of the usual one packet. The spacing between the first and the second packets is about 16 seconds to allow a modem call to complete, while the spacing between the remaining packets is about 2 seconds. This is designed to improve timekeeping quality with the server command and s addresses.
iburst	When the server is unreachable and at each poll interval, send a burst of eight packets instead of the usual one. As long as the server is unreachable, the spacing between packets is about 16 seconds to allow a modem call to complete. Once the server is reachable, the spacing between packets is about 2 seconds. This is designed to speed the initial synchronization acquisition with the server command and s addresses.
minpoll *interval*	Specifies the minimum polling interval for NTP messages, in seconds to the power of 2. The allowable range is 4 (16 seconds) to 14 (16384 seconds), inclusive. This option is not applicable to reference clocks. The default is 6 (64 seconds).
maxpoll *interval*	Specifies the maximum polling interval (in seconds), for NTP messages. The allowable range is 4 (16 seconds) to 14 (16384 seconds) inclusive. The default is 10 (1024 seconds). This option does not apply to reference clocks.
ttl *nn*	Specifies the time-to-live for multicast packets. Used only with broadcast and manycast modes.

- broadcastclient

  This statement enables reception of broadcast server messages to any local interface (type b) address. Upon receiving a message for the first time, the broadcast client measures the nominal server propagation delay using a brief client/server exchange with the server, then enters broadcastclient mode, in which it listens for and synchronizes to succeeding broadcast messages. Note that to avoid accidental or malicious disruption in this mode, both the server and client should use authentication and the same trusted key and key identifier.

- broadcastdelay *seconds*

  The broadcast and multicast modes require a special calibration to determine the network delay between the local and remote servers. Usually, this is done automatically by the initial protocol exchanges between the client and server. In some cases, the calibration procedure might fail possibly because of network or server access controls. This statement specifies the default delay to be used under these circumstances. Typically (for Ethernet), a number

between 0.003 and 0.007 seconds is appropriate. When this statement is not used, the default is 0.004 seconds.

- `multicastclient` *address*

  This statement enables reception of multicast server messages to the multicast group address(es) (type m) specified. Upon receiving a message for the first time, the multicast client measures the nominal server propagation delay using a brief client/server exchange with the server, then enters the broadcast client mode, in which it synchronizes to suceeding multicast messages.

  Note that to avoid accidental or malicious disruption in this mode, both the server and client should use authentication and the same trusted key and key identifier.

- `manycastserver` *address*

  This statement enables reception of manycast client messages to the multicast group address(es) (type m) specified. At least one address is required. The Internet Assigned Number Association (IANA) has assigned multicast group address 224.0.1.1 to NTP, but you should use this address only where the multicast span can be reliably constrained to protect neighbor networks. In general, you should use group addresses that have been given out by your administrator, as described in RFC 2365, or GLOP group addresses, as described in RFC 2770. Note that to avoid accidental or malicious disruption in this mode, both the server and client should use authentication and the same trusted key and key identifier.

- `driftfile` *file-specification*

  This statement specifies the name of the file used to record the frequency offset of the local clock oscillator. If the file exists, it is read at startup to set the initial frequency offset, and then is updated hourly with the current frequency computed by the NTP server.

  If the file does not exist or if the `driftfile` statement is not specified in the configuration file, the initial frequency offset is assumed to be zero. If the file does not exist but the `driftfile` keyword is specified without a parameter, the default, SYS$SPECIFIC:[TCPIP$NTP]TCPIP$NTP.DRIFT is used.

  In these cases, it might take some hours for the frequency to stabilize and for the residual timing errors to subside.

  The drift file TCPIP$NTP.DRIFT consists of a single floating-point number that records the frequency of the offset measured in parts per million (ppm).

- `enable auth | bclient | monitor | ntp | stats`

  `disable auth | bclient | monitor | ntp | stats`

  These statements enable and disable the following server options:

`auth`	Controls synchronization with unconfigured peers only if the peer has been correctly authenticated using a trusted key and key identifier. By default, `auth` is enabled.
`bclient`	Controls the server to listen for messages from broadcast or multicast servers. By default, `bclient` is disabled.
`monitor`	Controls the monitoring facility. By default, `monitor` is enabled.

ntp	Enables the server to adjust its local clock by means of NTP. If disabled, the local clock free runs at its intrinsic time and frequency offset. This statement is useful in case the local clock is controlled by some other device or protocol and NTP is used only to provide synchronization to other clients. In this case, the local clock driver can be used to provide this function and also certain time variables for error estimates and leap indicators. The default for this flag is enable.
stats	Enables the statistics facility. By default, stats is enabled.

- logconfig *configkeyword*

  This statement controls the amount and type of output written to the system log file. By default, all output is turned off. All configkeyword keywords can be prefixed with a plus sign (+) and a minus sign (-), where + adds messages and - removes messages. Messages can be controlled in four classes (clock, peer, sys, and sync). Within these classes, four types of messages can be controlled. Informational messages (info) control configuration information. Event messages (events) control logging of events (reachability, synchronization, alarm conditions). Statistics messages (statistics) control statistical output. The final message group is the status (status) messages. This message group describes mainly the synchronization status.

  Configuration keywords are formed by concatenating the message class with the event class. The all prefix can be used instead of a message class. A message class can also be followed by the all keyword to enable or disable all messages of the respective message class. Therefore, a minimal log configuration might look like the following example:

  ```
  logconfig +sysevents +syncstatus
  ```

  This configuration would list the synchronization state of the NTP server and the major system events.

  For a simple reference server, the following minimum message configuration might be useful:

  ```
  logconfig +syncall +clockall
  ```

  This configuration lists all clock information and synchronization information. All other events and messages about peers, system events, and so forth, are suppressed.

- tinker [panic *panic* | dispersion *dispersion* | minpoll *minpoll* | allan *allan* | huffpuff *huffpuff*]

  This statement can be used to alter several system variables in exceptional circumstances. It should occur in the configuration file before any other configuration options. The default values of these options have been carefully optimized for a wide range of network speeds and reliability expectations. In general, they interact in intricate ways that are hard to predict, and some combinations can result in unpredictable behavior. It is rarely necessary to change the default values.

  All options are in floating-point seconds or in seconds per second. The minpoll option is an integer in seconds to the power of 2. The options operate as follows:

  - panic *panic*

    This option becomes the new value for the panic threshold, normally 1000 seconds. If set to zero, the panic sanity check is disabled and a clock offset of any value is accepted.

- dispersion *dispersion*

  This option becomes the new value for the dispersion increase rate, usually .000015.

- minpoll *minpoll*

  This option becomes the new value for the minimum poll interval used when configuring a multicast client, a manycast client, and symmetric passive-mode association. The value defaults to 6 (64 seconds) and has a lower limit of 4 (16 seconds).

- allan *allan*

  This option becomes the new value for the minimum Allan intercept, which is a parameter of the PLL/FLL clock discipline algorithm. The value defaults to 1024 seconds, which is also the lower limit.

- huffpuff *huffpuff*

  This option becomes the new value for the experimental huff-n-puff filter span, which determines the most recent interval that the algorithm will search for a minimum delay. The lower limit is 900 seconds (15 minutes), but a more reasonable value is 7200 (2 hours). There is no default, since the filter is not enabled unless this statement is given.

B.3.2.1 NTP Monitoring Options

TCP/IP Services NTP includes a comprehensive monitoring facility that is suitable for continuous, long-term recording of server and client timekeeping performance. Statistics files are managed using file generation sets and scripts.

You can specify the following monitoring commands in your configuration file:

- statistics *name* [...]

  Enables writing of statistics records. The following is a list of the supported *name* statistics:

  - loopstats

    Enables recording of loop filter statistics information. Each update of the local clock outputs a line of the following form to the file generation set named loopstats:

    ```
    48773 10847.650 0.0001307 17.3478 2
    ```

    The first two fields show the date (Modified Julian Day) and time (seconds and fraction past UTC midnight). (A Julian Day [JD] begins at noon and runs until the next noon. The JD number is the number of days [or part of a day] since noon [UTC] on January 1, 4713 B.C. A Modified Julian Day [MJD] is the JD minus 2,400,000.5.)

    The next three fields show time offset (in seconds), frequency offset (in parts per million), and time constant of the clock discipline algorithm at each update of the clock.

  - peerstats

    Enables recording of peer statistics information. This includes statistics records of all peers of an NTP server and of special signals, where present and configured. Each valid update appends a line of the following form to the current element of a file generation set named peerstats:

    ```
    48773 10847.650 127.127.4.1 9714 -0.001605 0.00000 0.00142
    ```

The first two fields show the date (Modified Julian Day) and time (seconds and fraction past UTC midnight). The next two fields show the peer address in dotted-quad notation and status, respectively. The status field is encoded in hexadecimal in the format described in Appendix A of the NTP specification (RFC 1305). The final three fields show the offset, delay, and dispersion (in seconds).

- `clockstats`

  Enables recording of clock driver statistics information. Each update received from a clock driver outputs a line in the following form to the file generation set named `clockstats`:

  ```
  49213 525.624 127.127.4.1 93 226 00:08:29.606 D
  ```

  The first two fields show the date (Modified Julian Day) and time (seconds and fraction past UTC midnight). The next field shows the clock address in dotted-quad notation, The final field shows the last time code received from the clock in decoded ASCII format, where meaningful. In some clock drivers, a good deal of additional information can be gathered and displayed as well. For further details, see information specific to each clock.

- `rawstats`

  Enables recording of raw timestamps. Each valid update appends a line in the following form to the file generation set named `rawstats`:

  ```
  51554 79509.68  16.20.208.53  16.20.208.97
  3156617109.664603  3156617109.673268  3156617109.673268 31
  56617109.673268  3156617109.666556
  ```

  The first two fields show the date (Modified Julian Day) and time (seconds and fraction past UTC midnight). The next two fields show the peer and local addresses in dotted-quad notation. The next four fields are:

  * The originate timestamp
  * The received timestamp
  * The transmitted timestamp (the last one sent to the same peer)
  * The timestamp of the packet's arrival on the server

- `statsdir` *`directory-path`*

  Indicates the full path of a directory in which statistics files should be created.

B.3.2.2 Access Control Options

TCP/IP Services NTP implements a general-purpose address-and-mask based restriction list. The list is sorted by address and by mask, and the list is searched in this order for matches. The last match to be found defines the restriction flags associated with the incoming packets. The source address of incoming packets is used for the match. The 32-bit address is and'ed with the mask associated with the restriction entry, and then is compared with the entry's address (which has also been and'ed with the mask) to look for a match.

Although this facility might be useful for keeping unwanted or broken remote time servers from affecting your own, it is not considered an alternative to the standard NTP authentication facility.

B.3.2.2.1 The Kiss-of-Death Packet Ordinarily, packets denied service are simply dropped with no further action except to increment statistics counters. Sometimes a more proactive response is needed, such as a server message that explicitly requests the client to stop sending and leave a message for the system operator. A special packet format has been created for this purpose called the kiss-of-death (kod) packet. If the `kod` flag is set and either service is denied or the client limit is exceeded, the server returns the packet and sets the leap bits unsynchronized, stratum 0, and the ASCII string "DENY" in the reference source identifier field. If the `kod` flag is not set, the server simply drops the packet.

A client or peer that receives a kiss-of-death packet performs a set of sanity checks to minimize security exposure. If this is the first packet received from the server, the client assumes an access-denied condition at the server. The client updates the stratum and reference identifier peer variables and sets the access-denied bit in the peer `flash` variable (for information about displaying the `flash` variable, see Section B.8.1.2). If this bit is set, the client sends no packets to the server. If this is not the first packet, the client assumes a client limit condition at the server but does not update the peer variables. In either case, a message is sent to the server's log file.

B.3.2.2.2 Access Control Statements and Flags The syntax for the restrict statement is as follows:

- `restrict` *address* `[mask` *mask*`]` `[`*flag*`][...]`

  The *address* argument, expressed in dotted-quad form, is the address of a host or network. The *mask* argument, also expressed in dotted-quad form, defaults to 255.255.255.255, meaning that *address* is treated as the address of an individual host. A default entry (address 0.0.0.0, mask 0.0.0.0) is always included and, given the sort algorithm, is always the first entry in the list. Note that, while *address* is normally given in dotted-quad format, the text string *default*, with no *mask* option, can be used to indicate the default entry.

  Flag always restricts access (that is, an entry with no flags indicates that free access to the server is to be given). The flags are not orthogonal in that more restrictive flags often make less restrictive ones redundant. The flags can generally be classed into two categories: those that restrict time service and those that restrict informational queries and attempts to do run-time reconfiguration of the server.

  You can specify one or more of the flags shown in the following table:

Table B–1 Restrict Statement Flags

Flag	Description
`kod`	If access is denied, send a kiss-of-death packet.
`ignore`	Ignore all packets from hosts that match this entry. If this flag is specified, neither queries nor time server polls will be responded to.
`noquery`	Ignore all NTP mode 6 and 7 packets (that is, information queries and configuration requests, respectively) from the source. Time service is not affected.

(continued on next page)

Table B–1 (Cont.) Restrict Statement Flags

Flag	Description
nomodify	Ignore all NTP mode 6 and 7 packets that attempt to modify the state of the server (that is, run time reconfiguration). Queries that return information are permitted.
noserve	Ignore NTP packets whose mode is other than 6 or 7. In effect, time service is denied, though queries are still permitted.
nopeer	Provide stateless time service to polling hosts, but do not allocate peer memory resources to these hosts even if they might be considered useful as future synchronization partners.
notrust	Treat these hosts normally in other respects, but never use them as synchronization sources.
limited	These hosts are subject to limitation of number of clients from the same net. In this context, net refers to the IP notion of net (class A, class B, class C, and so forth). Only the first clientlimit hosts that have shown up at the server and that have been active during the last clientperiod seconds are accepted. Requests from other clients from the same net are rejected; only time request packets are taken into account. Query packets sent by the NTPQ and NTPDC programs are not subject to these limits. A history of clients is kept using the monitoring capability of the NTP server. Thus, monitoring is always active as long as there is a restriction entry with the limited flag.
ntpport	This is actually a match algorithm modifier, rather than a restriction flag. Its presence causes the restriction entry to be matched only if the source port in the packet is the standard NTP UDP port (123). Both ntpport and non-ntpport can be specified. The ntpport is considered more specific and is sorted later in the list.
version	Ignore these hosts if not the current NTP version. Default restriction list entries, with the flags ignore, interface, ntpport, for each of the local host's interface addresses are inserted into the table at startup to prevent the server from attempting to synchronize to its own time. A default entry is also always present if it is otherwise unconfigured. No flags are associated with the default entry (that is, everything besides your own NTP server is unrestricted).

- clientlimit *limit*

  The *limit* sets the client_limit variable, which limits the number of simultaneous access-controlled clients. The default value for this variable is 3.

- clientperiod *period*

  The *period* sets the client_limit_period variable that specifies the number of seconds after which a client is considered inactive and thus no longer counted for client limit restriction. The default value for this variable is 3600 seconds.

B.3.2.3 Sample NTP Configuration File

A sample of the NTP configuration template follows:

```
#             Copyright 2000 Compaq Computer Corporation
#
#                  Example NTP Configuration File
#
# Rename this template to TCPIP$NTP.CONF.
#
# See the Compaq TCP/IP Services for OpenVMS Management manual for
# additional commands and detailed instructions on using this
# configuration file.
#
# The Network Time Protocol (NTP) provides synchronized timekeeping among
# a set of distributed time servers and clients. The local OpenVMS host
# maintains an NTP configuration file, TCPIP$NTP.CONF, of participating peers.
# TCPIP$NTP.CONF is maintained in the SYS$SPECIFIC:[TCPIP$NTP] directory.
#
# As the system manager populating this file, you must determine the
# peer hosts with which the local hosts should negotiate and synchronize.
# Include at least one (but preferably three) hosts that you are
# certain have the following characteristics:
#
#       * provide accurate time
#       * synchronize to Internet Time Servers (if they are not themselves
#         Internet Time Servers)
#
# The NTP configuration file is not dynamic, and therefore requires
# restarting NTP  after being edited to make the changes take effect.
# However, you can make run-time configuration requests interactively
# using the TCPIP$NTPDC utility.

# Your NTP configuration file should always include the following
# driftfile entry.  The driftfile is the name of the file that stores
# the clock drift (also known as frequency error) of the system clock.

driftfile SYS$SPECIFIC:[TCPIP$NTP]TCPIP$NTP.DRIFT

# Sample peer entries follow.  Replace them with your own list of hosts
# and identify the appropriate association mode.  If you specify
# multiple hosts, NTP can choose the best source with which to
# synchronize. This also provides reliability in case one of the hosts
# becomes unavailable.

# Identify each peer with a fully qualified DNS host name or with
# an IP address in dotted-quad notation.

peer 18.72.0.3
peer 130.43.2.2
peer 16.1.0.22
peer parrot

# The following commands allow interoperation of NTP with another time service
# such as DTSS.  If enabled (by removing #), NTP will not set the system clock.

# server 127.127.1.0 prefer
# fudge 127.127.1.0 stratum 0

# The following commands allow this node to act as a backup NTP server (or as
# the sole NTP server on an isolated network), using its own system clock as
# the reference source.  If enabled (by removing #), this NTP server will
# become active only when all other normal synchronization sources are
# unavailable.

# server 127.127.1.0
# fudge 127.127.1.0 stratum 8
```

B.3.3 Using NTP with Another Time Service

A local host can run more than one time service. For example, a host can have both NTP and DTSS (Digital Time Synchronization Service) installed. However, only one of these time services is allowed to set the system clock.

If you are running a time service in addition to NTP, you must stop either the other time source or NTP from setting the system clock. You can stop NTP from setting the system clock by adding the following statements to the configuration file:

```
server 127.127.1.0 prefer
fudge 127.127.1.0 stratum 0
```

In these statements, the hardware address of the local clock (LOCAL) is 127.127.1.0. These statements force NTP to use its own system clock as a reference clock. The host continues to respond to NTP time queries but does not make any adjustments to the system clock, thereby allowing the other time service to make those changes.

B.4 Configuring NTP as Backup Time Server

You can configure the NTP service as a backup time server. In this case, if all other network synchronization sources become unavailable, the NTP service becomes active. You can also use this method to allow the local node to act as the NTP server in an an isolated network. To configure the NTP service as the backup server or the sole NTP server, enter the following commands in the NTP configuration file:

```
server 127.127.1.0
fudge 127.127.1.0 stratum 8
```

In this example, the stratum is set to a high number (8) so that it will not interfere with any other, possibly better, time synchronization source. You should set the stratum to a number that is higher than the stratum of all other time synchronization sources.

B.5 NTP Event Logging

NTP maintains a record of system clock updates in the file SYS$SPECIFIC:[TCPIP$NTP]TCPIP$NTP_RUN.LOG. NTP reopens this log file daily, each time creating a new version of the file (older versions are not automatically purged). Events logged to this file can include the following messages:

- Synchronization status that indicates synchronization was lost, stratum changes, and so forth
- System time adjustments
- Time adjustment status

Logging can be increased by using the logconfig option in TCPIP$NTP.CONF. For more information, see Section B.3.2.

In addition, you can enable debugging diagnostics by defining the following logical name with /SYSTEM and a value from 1 through 6, where 6 specifies the most detailed logging:

```
$ DEFINE /SYSTEM TCPIP$NTP_LOG_LEVEL n
```

Table B–2 describes the messages most frequently included in the NTP log file.

Table B–2 NTP Log File Messages

Message	Description
Time slew *time*	Indicates that NTP has set the local clock by slewing the local time to match the synchronization source. This happens because the local host is no longer synchronized. For example: `time slew -0.218843 s`
Synchronization lost	This usually occurs after a time reset. All peer filter registers are cleared, for example, for that particular peer; all state variables are reset along with the polling interval; and the clock selection procedure is once again performed.
Couldn't resolve *hostname*, giving up on it	Indicates that the host name could not be resolved. This peer will not be considered for the candidate list of peers. For example: `couldn't resolve 'fred',  giving up on it`
Send to *IP-address*: *reason*	Indicates that a problem occurred while sending a packet to its destination. The most common reason logged is "connection refused." For example: `sendto(16.20.208.100): connection refused`
Time Correction of *delta-time* seconds exceeds sanity limit (1000); set clock manually to the correct UTC time	NTP has detected a time difference greater than 1000 seconds between the local clock and the server clock. You must set the clock manually or use the NTPDATE program and then restart NTP. Once NTP sets the clock, it continuously tracks the discrepancy between the local time and NTP time and adjusts the clock accordingly.
offset: *n* sec freq *x* ppm poll: *y* sec error *z*	An hourly message, in which: • `offset` is the offset (in seconds) of the peer clock relative to the local clock (that is, the amount to adjust the local clock to bring it into correspondence with the reference clock). • `freq` is the computed error in the intrinsic frequency of the local clock (also known as "drift") (in parts per million). • `poll` is the minimum interval (in seconds) between transmitted messages (that is, messages sent between NTP peers, as in a client to a server). • `error` is the measure of network jitter (that is, latencies in computer hardware and software).

(continued on next page)

Table B–2 (Cont.) NTP Log File Messages

Message	Description
No clock adjustments will be made, DTSS is active	Indicates that the DTSS time service is running on the system. The DTSS time service should be disabled if you would like NTP to set the system time. To disable the DTSS time service, enter the following command: `$ RUN SYS$SYSTEM:NCL DISABLE DTSS` Alternatively, you can configure the NTP server not to make clock adjustments, as described in Section B.3.3. NTP dynamically detects whether the DTSS time service is enabled at any time and will log this message if appropriate.
Clock adjustments will resume. DTSS no longer active	Indicates that the DTSS time service has been disabled on the system. NTP will now handle clock adjustments. NTP dynamically detects whether the DTSS time service is enabled at any time and will log this message if appropriate.

B.5.1 Sample NTP Log Files

The following sample shows a standard NTP log file that has no extra logging enabled.

```
2 Jul 15:33:37  ntpd version = 4.1.0
2 Jul 15:33:37  precision = 976 usec
2 Jul 15:33:37  frequency initialized -66.795 from SYS$SPECIFIC:[TCPIP$NTP]
  TCPI P$NTP.DRIFT
2 Jul 15:37:01  time slew 0.148981 s
2 Jul 16:33:38  offset: 0.008022 sec  freq: 1.301 ppm  poll: 128 sec  error:
  0.014056
2 Jul 17:33:41  offset: 0.003190 sec  freq: 4.218 ppm  poll: 256 sec  error:
  0.007071
2 Jul 18:33:41  offset: -0.000622 sec  freq: 4.575 ppm  poll: 512 sec  error:
  0.005358
2 Jul 19:33:41  offset: -0.003216 sec  freq: 3.749 ppm  poll: 1024 sec  error:
  0.005610
2 Jul 20:33:41  offset: -0.000899 sec  freq: 2.823 ppm  poll: 1024 sec  error:
  0.005710
2 Jul 21:33:41  offset: -0.000299 sec  freq: 2.510 ppm  poll: 1024 sec  error:
  0.005468
2 Jul 22:08:04  time slew -0.156010 s
2 Jul 22:33:41  offset: 0.002615 sec  freq: 4.022 ppm  poll: 1024 sec  error:
  0.005297
2 Jul 23:33:41  offset: -0.002466 sec  freq: 3.237 ppm  poll: 1024 sec  error:
  0.005626
3 Jul 00:33:41  offset: 0.000100 sec  freq: 1.737 ppm  poll: 1024 sec  error:
  0.006343
3 Jul 01:33:41  offset: 0.002842 sec  freq: 2.393 ppm  poll: 1024 sec  error:
  0.006023
3 Jul 02:33:41  offset: 0.000089 sec  freq: 3.204 ppm  poll: 1024 sec  error:
  0.006199
3 Jul 03:33:41  offset: 0.001094 sec  freq: 3.576 ppm  poll: 1024 sec  error:
  0.005628
```

The next sample shows an NTP log file with all categories of logging enabled.

```
10 Jul 13:38:05  ntpd version = 4.1.0
10 Jul 13:38:05  precision = 976 usec
10 Jul 13:38:05  frequency initialized 3.157 from SYS$SPECIFIC:[TCPIP$NTP]TCPIP$
NTP.DRIFT
10 Jul 13:38:05  system event 'event_restart' (0x01) status 'sync_alarm, sync_un
spec, 1 event, event_unspec' (0xc010)
10 Jul 13:38:11  peer 204.123.2.70 event 'event_reach' (0x84) status 'unreach, c
onf, 1 event, event_reach' (0x8014)
10 Jul 13:38:20  peer 204.123.2.71 event 'event_reach' (0x84) status 'unreach, c
onf, 1 event, event_reach' (0x8014)
10 Jul 13:38:22  peer 16.140.0.12 event 'event_reach' (0x84) status 'unreach, co
nf, 1 event, event_reach' (0x8014)
10 Jul 13:39:40  system event 'event_peer/strat_chg' (0x04) status 'sync_alarm,
sync_ntp, 2 events, event_restart' (0xc621)
10 Jul 13:39:49  system event 'event_sync_chg' (0x03) status 'leap_none, sync_nt
p, 3 events, event_peer/strat_chg' (0x634)
10 Jul 13:39:49  system event 'event_peer/strat_chg' (0x04) status 'leap_none, s
ync_ntp, 4 events, event_sync_chg' (0x643)
10 Jul 13:51:24  peer 16.141.40.135 event 'event_reach' (0x84) status 'unreach,
conf, 1 event, event_reach' (0x8014)
10 Jul 14:02:08  peer 16.141.40.135 event 'event_unreach' (0x83) status 'unreach
, conf, 2 events, event_unreach' (0x8023)
10 Jul 14:12:47  peer 16.141.40.135 event 'event_reach' (0x84) status 'unreach,
conf, 3 events, event_reach' (0x8034)
10 Jul 14:38:06  offset: 0.015558 sec  freq: 4.407 ppm  poll: 128 sec  error: 0.
008575
10 Jul 14:45:54  peer 16.141.40.135 event 'event_unreach' (0x83) status 'unreach
, conf, 4 events, event_unreach' (0x8043)
10 Jul 15:38:07  offset: 0.021501 sec  freq: 8.734 ppm  poll: 512 sec  error: 0.
015413
10 Jul 15:44:47  peer 16.141.40.135 event 'event_reach' (0x84) status 'unreach,
conf, 5 events, event_reach' (0x8054)
10 Jul 16:38:07  offset: 0.016173 sec  freq: 25.014 ppm  poll: 1024 sec  error:
0.011453
10 Jul 17:38:07  offset: -0.043169 sec  freq: 13.291 ppm  poll: 1024 sec  error:
 0.024752
10 Jul 18:38:07  offset: -0.017786 sec  freq: 6.005 ppm  poll: 1024 sec  error:
0.025309
```

B.6 NTP Authentication Support

Authentication support is implemented using the MD5 algorithm to compute a message digest. The servers involved in an association must agree on the key and key identifier used to authenticate their messages.

Keys and related information are specified in a key file. Keys are used for:

- Ordinary NTP associations
- The NTPQ utility program
- The NTPDC utility program

B.6.1 NTP Authentication Commands

Table B–3 describes additional configuration statements and options that support authentication.

Table B–3 Authentication Commands

Command	Description
`keys` *keys-file*	Specifies the file name for the keys file, which contains the encryption keys and key identifiers used by NTP, NTPQ, and NTPDC when operating in authenticated mode.
`trustedkey` *key-ID* [...]	Specifies the encryption key identifiers that are trusted for the purposes of authenticating peers suitable for synchronization, as well as keys used by the NTPQ and NTPDC programs. The authentication procedures require that the local and remote servers share the same key-ID and key value for this purpose, although different key values can be used with different servers. The *key-ID* arguments are 32-bit unsigned decimal integers from 1 to 15. Note that the NTP key 0 is used to indicate an invalid key value or key identifier; therefore, it should not be used for any other purpose.
`requestkey` *key-ID*	Specifies the key identifier to use with the NTPDC program, which uses a proprietary protocol specific to this implementation of NTP. This program is useful in diagnosing and repairing problems that affect the operation of NTP. For information about NTPDC, see Section B.7.3. The *key-ID* argument to this command is an unsigned 32-bit decimal number that identifies the trusted key in the keys file. If the `requestkey` command is not included in the configuration file, or if the keys do not match, any request to change a server variable is denied.
`controlkey` *key-ID*	Specifies the key identifier to use with the NTPQ program, which uses the standard protocol defined in RFC 1305. This program is useful in diagnosing and repairing problems that affect the operation of NTP. For more information about NTPQ, see Section B.7.4. The *key-ID* argument to this command is a 32-bit decimal integer that identifies a trusted key in the keys file. If the `controlkey` command is not included in the configuration file, or if the keys do not match, any request to change a server variable is denied.

Keys are defined in a keys file, as described in Section B.6.2.

B.6.2 Authentication Key Format

The NTP service reads keys from a keys file that is specified using the `keys` command in the configuration file. You can supply one or more keys from 1 to 15 in the keys file.

Key entries use the following format:

```
key-ID key-type key-value
```

The fields include:

- *key-ID*, which is an arbitrary, unsigned 32-bit number (in decimal). The range of possible values is 1 to 15. Key IDs are specified by the `requestkey` and `controlkey` statements in the configuration file. The key ID number 0 (56 zero bits) is reserved; it is used to indicate an invalid key ID or key value.
- *key-type*, which identifies the type of key value. Only one key format, M, is currently supported. This indicates that the MD5 authentication scheme is being used.
- *key-value*, which is an ASCII string from one to eight characters. The following characters are not allowed:

  space
  pound sign (#)
  \ t
  \ n
  \ 0

Because this file contains authorization data, Compaq recommends that you limit read access to this file. In particular, you should disable world read access.

The following is a sample keys file:

```
#
#
4       M    DonTTelL
6       M    hElloWrl
12      M    ImASecrt
```

B.7 NTP Utilities

NTP provides several utility programs that help you manage and make changes to the NTP server. These utilities include:

- NTPDATE, the date and time utility that sets the local date and time by polling the specified server. Run NTPDATE manually or from the host startup script to set the clock at boot time before NTP starts.

  NTPDATE does not set the date if NTP is already running on the same host.

  For information about using NTPDATE, see Section B.7.1.
- NTPTRACE, the trace utility that follows the chain of NTP servers back to their master time source. For information about using NTPTRACE, see Section B.7.2.
- NTPDC, the special query program that provides extensive state and statistics information and allows you to set configuration options at run time. Run this program in interactive mode or with command line arguments.

  For information about using NTPDC, see Section B.7.3.
- NTPQ, the standard query program that queries NTP servers about their current state and requests changes to that state.

  For information about using NTPQ, see Section B.7.4.
- NTP_GENKEYS, the random key generator program that generates random keys that are used by the NTP Version 3 and NTP Version 4 symmetric key authentication scheme.

  For information about using NTP_GENKEYS, see Section B.7.5.

To define the commands described in the following sections, run the following procedure:

```
$ @SYS$MANAGER:TCPIP$DEFINE_COMMANDS.COM
```

B.7.1 Setting the Date and Time with NTPDATE

The NTPDATE program sets the local date and time by polling a specified server or servers to determine the correct time. A number of samples are obtained from each of the servers specified, and a subset of the NTP clock filter and selection algorithms are applied to select the best samples. The accuracy and reliability of NTPDATE depends on the number of servers it polls, the number of polls it makes each time it runs, and the interval length between runs.

Run NTPDATE manually to set the host clock or from the host startup file to set the clock at boot time. In some cases, it is useful to set the clock manually before you start NTP. The NTPDATE program makes time adjustments (called "stepping the time") by calling the OpenVMS routine SYS$SETIME.

Note

NTPDATE does not set the date and time if an NTP server is running on the same host.

Enter specific commands using the following format:

```
NTPDATE [option...] host [host...]
```

For example, the following command sets the clock based on the time provided from one of the specified hosts (BIRDY, OWL, or FRED):

```
$ NTPDATE BIRDY OWL FRED
```

NTP sets the date and time by polling the servers you specify as arguments to the command. Samples are obtained from each of the specified servers. NTP then analyzes the results to select the best server to use as a time source. Table B–4 describes the NTPDATE command options.

Table B–4 NTPDATE Options

Option	Description
-d	Changes the time and prints information useful for debugging.
-o *version*	Specifies the NTP version (1, 2, or 3) for outgoing packets (for compatibility with older versions of NTP). Version 4 is the default.
-p *n*	Specifies the number of samples NTPDATE acquires from each server. The default is 4. You can specify from 1 to 8.
-q	Specifies a query only; does not set the clock.

B.7.2 Tracing a Time Source with NTPTRACE

Use the NTPTRACE utility to determine the source from which an NTP server obtains its time. NTPTRACE follows the chain of time servers back to the master time source.

Use the following syntax when entering commands:

```
NTPTRACE [option...]
```

The following example shows output from an NTPTRACE command. In the following example, the chain of servers is from the local host to the stratum 1 server FRED, which is synchronizing to a GPS reference clock.

```
$ NTPTRACE

LOCALHOST: stratum 3, offset -0.000000, synch distance1.50948
parrot.birds.com: stratum 2, offset -0.126774, synch distance 0.00909
fred.birds.com: stratum 1, offset -0.129567, synch distance 0.00168,
refid 'GPS'
```

All times are in seconds. The output fields on each line are as follows:

- Host name
- Host stratum
- Time offset between the host and the local host (not always zero for LOCALHOST).
- Synchronization distance
- Reference clock ID (only for stratum 1 servers)

Table B–5 describes the NTPTRACE command options.

Table B–5 NTPTRACE Options

Option	Description
-d	Enables debugging output.
-n	Displays IP addresses instead of host names. This may be necessary if a name server is down.
-r *retries*	Sets the number of retransmission attempts for each host. The default is 5.
-t *timeout*	Sets the retransmission timeout (in seconds). The default is 2.
-v	Displays additional information about the NTP servers.

B.7.3 Making Run-Time Requests with NTPDC

You can make run-time changes to NTP with query commands by running the NTPDC utility. NTPDC displays time values in seconds.

Run-time requests are always authenticated requests. Authentication not only provides verification that the requester has permission to make such changes, but also gives an extra degree of protection against transmission errors.

The reconfiguration facility works well with a server on the local host and between time-synchronized hosts on the same LAN. The facility works poorly for more distant hosts. Authenticated requests include a timestamp. The server compares the timestamp to its *receive* timestamp. If they differ by more than a small amount, the request is rejected for the following reasons:

- To make it more difficult for an intruder to overhear traffic on your LAN.
- To make it more difficult for topologically remote hosts to request configuration changes to your server.

To run NTPDC, enter the following command:

```
$ NTPDC
NTPDC>
```

At the NTPDC> prompt, enter the appropriate type of command from the following list:

- Interactive commands
- Control commands
- Run-time configuration request commands

The following sections describe the NTPDC commands.

B.7.3.1 NTPDC Interactive Commands

Interactive commands consist of a command name followed by one or more keywords. The interactive commands include:

- `help [`*`command-keyword`*`]`

  Enter a question mark (?) to display a list of all the command keywords known to this version of NTPDC. Enter a question mark followed by a command keyword to display information about the function and use of the command.

- `host` *`hostname`*

  Sets the host to which future queries will be sent. The *hostname* can be either a host name or a numeric address.

- `hostnames [ yes | no ]`

  If you specify `yes`, host names are displayed. If you specify `no`, numeric addresses are displayed. The default is `yes` unless you include the `-n` option on the command line, as described in Table B–5.

- `keyid` *`key-ID`*

  Specifies the key number to be used to authenticate configuration requests. This must correspond to a key number the server has been configured to use for this purpose.

- `quit`

  Exits NTPDC.

- `passwd`

  Prompts you to type in a password (not echoed) that will be used to authenticate configuration requests. The password must correspond to the key configured for use by the NTP server for this purpose.

- `timeout` *`milliseconds`*

  Specify a timeout period for responses to server queries. The default is about 8000 milliseconds (8 seconds). Because NTPDC retries each query once after a timeout, the total waiting time for a timeout will be twice the timeout value set.

B.7.3.2 NTPDC Control Message Commands

Control message commands request information about the server. These are read-only commands in that they make no modification of the server configuration state.

The NTPDC control message commands include:

- `listpeers`

Displays a brief list of the peers for which the server is maintaining state. These include all configured peer associations as well as those peers whose stratum is such that the server considers them to be possible future synchronization candidates.

- `peers`

  Obtains a list of peers for which the server is maintaining state, along with a summary of that state. The summary information includes:

  - The address of the remote peer
  - The local interface address (0.0.0.0 if a local address has not been determined)
  - The stratum of the remote peer (a stratum of 16 indicates the remote peer is unsynchronized)
  - The polling interval (in seconds)
  - The reachability register (in octal)
  - The current estimated delay, offset, and dispersion of the peer (in seconds)

  In addition, the character in the left margin indicates the operating mode of this peer entry, as follows:

  Plus sign (+) denotes symmetric active.
  Minus sign (-) indicates symmetric passive.
  Equals sign (=) means the remote server is being polled in client mode.
  Up arrow (^) indicates that the server is broadcasting to this address.
  Tilde (~) denotes that the remote peer is sending broadcasts.
  Asterisk (*) marks the peer to which the server is currently synchronizing.

  The contents of the host field can be one of the following four forms:

  - Host name
  - IP address
  - Reference clock implementation name with its parameter
  - `REFCLK (`*`implementation number parameter`*`)`

  If you specify `hostnames no`, only IP addresses are displayed.

- `dmpeers`

  Displays a slightly different peer summary list, identical to the output of the `peers` command except for the character in the leftmost column. Characters appear only beside peers that were included in the final stage of the clock selection algorithm:

  Dot (.) indicates that this peer was rejected in the falseticker detection.
  Plus sign (+) indicates that the peer was accepted.
  Asterisk (*) denotes the peer to which the server is currently synchronizing.

- `showpeer` *`peer-address`* `[...]`

  Shows a detailed display of the current peer variables for one or more peers.

- `pstats` *`peer-address`* `[...]`

  Shows per-peer statistics counters associated with the specified peers.

- loopinfo [oneline | multiline]

  Displays the values of selected loop-filter variables. The loop filter is the part of NTP that adjusts the local system clock. These options include:

  - offset — the last offset given to the loop filter by the packet processing code.
  - frequency — the frequency error of the local clock (in parts per million)
  - time_const — controls the stiffness of the phase-lock loop and, therefore, the speed at which it can adapt to oscillator drift.
  - watchdog timer value — the number of seconds that have elapsed since the last sample offset was given to the loop filter.

  The oneline and multiline options specify the format in which this information is to be displayed; multiline is the default.

- sysinfo

  Displays a variety of system state variables, such as the state related to the local server. These variables include:

  - system flags — shows various system flags, some of which can be set and cleared by the enable and disable configuration commands, respectively. These are the auth, bclient, monitor, ntp, and stats flags.
  - stability — the residual frequency error remaining after the system frequency correction is applied. It is intended for maintenance and debugging.
  - broadcastdelay — shows the default broadcast delay as set by the broadcastdelay configuration command.
  - authdelay — shows the default authentication delay as set by the authdelay configuration command.

- sysstats

  Displays statistics counters maintained in the protocol module.

- memstats

  Displays statistics counters related to memory allocation code.

- iostats

  Displays statistics counters maintained in the input/output module.

- timerstats

  Displays statistics counters maintained in the timer/event queue support code.

- reslist

  Displays the server's restriction list. This list is displayed in the order in which the restrictions are applied.

- monlist [*version*]

  Displays traffic counts collected. This information is maintained by the monitor facility. Normally, you should not need to specify the version number.

B.7.3.3 NTPDC Request Commands

The following commands make authenticated requests:

- addpeer *peer-address key-ID* [*version*] [*prefer*]

  Adds a configured peer association at the given address and operates in symmetric active mode. The existing association with the same peer can be deleted when this command is executed or can be converted to conform to the new configuration.

  The *key-ID* is the key identifier for requestkey, as described in Table B–3. All outgoing packets to the remote server will have an authentication field attached that is encrypted with this key.

  The value for *version* can be 1, 2, 3 or 4. The default is Version 4.

  The *prefer* keyword indicates a preferred peer that will be used for clock synchronization, if possible.

- addserver *peer-address key-ID* [*version*] [*prefer*]

  This command is the same as addpeer except that the operating mode is client.

- broadcast *peer-address key-ID* [*version*] [*prefer*]

  This command is the same as addpeer except that the operating mode is broadcast. In this case, a valid key identifier and key value are required. The *peer-address* parameter can be either the broadcast address of the local network or a multicast group address assigned to NTP.

- unconfig *peer-address* [...]

  Causes the configured bit to be removed from the specified remote peer. This deletes the peer association. When appropriate, however, the association may persist in an unconfigured mode if the remote peer is willing to continue in this fashion.

- enable [*flag*] [...]

  disable [*flag*] [...]

  These commands operate in the same way as the enable and disable configuration commands. For details, see Section B.3.2.

- fudge *peer-address* [*time1*] [*time2*] [stratum *stratum*] [*refID*]

  Provides a way to set time, stratum, and identification data for a reference clock. (The TCP/IP Services product supports only the local reference clock.)

Use the following syntax to enter the NTPDC foreign command:

```
NTPDC [-i] [-l] [-n] [-p] [-s] [-c command][host1,host2,...]
```

Table B–6 describes the NTPDC options.

Table B–6 NTPDC Options

Option	Description
-c *command*	The command argument is interpreted as an interactive format command and is added to the list of commands to be executed on the specified hosts. You can specify multiple -c options.
-i	Forces NTPDC to operate in interactive mode.
-l	Obtains a list of peers that are known to the servers.
-n	Displays all host addresses in numeric format rather than convert them to host names.
-p	Displays a list of the peers known to the server as well as a summary of their state.
-s	Displays a list of the peers known to the server as well as a summary of their state. Uses a slightly different format than the -p option.

B.7.4 Querying the NTP Server with NTPQ

The NTPQ program allows you to query the NTP server about its current state and to request changes to that state. NTPQ can also obtain and display a list of peers in a common format by sending multiple queries to the server.

The NTPQ program authenticates requests based on the key entry in the keys file that is configured using the controlkey command (described in Table B–3).

The NTPQ program uses NTP mode 6 packets to communicate with the NTP server; therefore, NTPQ can query any compatible server on the network. Because NTP is a UDP protocol, this communication is somewhat unreliable over long distances (in terms of network topology). The NTPQ program makes one attempt to restransmit requests and times out requests if the remote host does not respond within the expected amount of time. NTPQ displays time values in milliseconds.

To run the NTPQ program, enter the following command:

```
$ NTPQ
NTPQ>
```

At the NTPQ> prompt, enter commands in the following syntax:

command [*options...*]

The following commands allow you to query and set NTP server state information:

- ? [*command-keyword*]

  A question mark (?) by itself prints a list of all the command keywords known to this version of NTPQ. A question mark followed by a command keyword prints function and usage information about the command.

- addvars *variable-name*[=*value*] [,...]
- rmvars *variable-name* [,...]
- clearvars

  The data carried by NTP mode 6 messages consists of a list of items in the following form:

  variable-name=value

In requests to the server to read variables, the =*value* portion is ignored and can be omitted. The NTPQ program maintains an internal list in which data to be included in control messages can be assembled and sent using the `readlist` and `writelist` commands. The `addvars` command allows variables and their optional values to be added to the list. If you want to add more than one variable, separate the list items by commas and do not include blank spaces. The `rmvars` command removes individual variables from the list, while the `clearvars` command removes all variables from the list.

- `authenticate yes | no`

  By default, NTPQ does not authenticate requests unless they are write requests. The `authenticate yes` command causes NTPQ to send authentication with all requests it makes. Authenticated requests cause some servers to handle requests slightly differently. To prevent any mishap, do a peer display before turning on authentication.

- `cooked`

  Reformats variables that are recognized by the server. Variables that NTPQ does not recognize are marked with a trailing question mark (?).

- `debug more | less | no`

  Adjusts the level of NTPQ debugging. The default is `debug no`.

- `help`

  Displays the list of NTPQ interactive commands. This is the same as question mark (?).

- `host [`*host-name*`]`

  Sets the host to which future queries will be sent; *host-name* can be either a host name or an Internet address. If *host-name* is not specified, the current host is used.

- `hostnames yes | no`

  If `yes` is specified, displays host names in information displays. If `no` is specified, displays Internet addresses instead. The default is `hostnames yes`. The default can be modified using the command line option `-n`.

- `key-ID` *n*

  Specifies the key ID number to be used to authenticate configuration requests. This must correspond to a key ID number the server has been configured to use for this purpose.

- `keytype md5 | des`

  Sets the authentication key to either MD5 or DES. Only MD5 is supported in this implementation.

- `ntpversion 1 | 2 | 3 | 4`

  Sets the NTP version number that NTPQ claims in packets. Default is 4. Mode 6 control messages (as well as modes) did not exist in NTP Version 1.

- `passwd`

  Prompts you to enter a password (not echoed) that is used to authenticate configuration requests. The password must correspond to the key value configured for use by the NTP server for this purpose (see Section B.6.2).

- `quit`

  Exits NTPQ.

- `raw`

  Displays all output from query commands as received from the remote server. The only data formatting performed is to translate non-ASCII data into a printable form.

- `timeout` *`milliseconds`*

  Specifies a timeout period for responses to server queries. The default is about 5000 milliseconds. Since NTPQ retries each query once after a timeout, the total waiting time for a timeout will be twice the timeout value.

B.7.4.1 NTPQ Control Message Commands

Each peer known to an NTP server has a 16-bit integer association identifier assigned to it. NTP control messages that carry peer variables must identify the peer that the values correspond to by including the peer's association ID. An association ID of zero indicates the variables are system variables whose names are drawn from a separate name space.

Control message commands result in one or more NTP mode 6 messages being sent to the server, and cause the data returned to be displayed in a format that you control using the commands listed in Section B.7.4. Most control message commands send a single message and expect a single response. The exceptions are the `peers` command, which sends a preprogrammed series of messages to obtain the data it needs, and the `mreadlist` and `mreadvar` commands, which are repeated for each specified association.

- `associations`

  Displays a list of association identifiers and peer status for recognized peers of the server being queried. The list is printed in columns. The first of these is an index numbering the associations from 1 for internal use; the second is the actual association identifier returned by the server; and the third is the status word for the peer. This is followed by a number of columns containing data decoded from the status word. The data returned by the associations command is cached internally in NTPQ. The index is then used when dealing with servers that use association identifiers. For any subsequent commands that require an association identifier as an argument, the index can be used as an alternative.

- `lassociations`

  Obtains and displays a list of association identifiers and peer status for all associations for which the server is maintaining state. This command differs from the `associations` command only for servers that retain state for out-of-spec client associations. Such associations are normally omitted from the display when the `associations` command is used but are included in the output of the `lassociations` command.

- lopeers

  Obtains and displays a list of all peers and clients having the destination address.

- lpassociations

  Displays data for all associations, including unrecognized client associations, from the internally cached list of associations.

- lpeers

  Similar to peers except that a summary of all associations for which the server is maintaining state is displayed. This command can produce a much longer list of peers.

- mreadlist *assocID assocID*

  Similar to the readlist command except that the query is done for each of a range of (nonzero) association IDs. This range is determined from the association list cached by the most recent associations command.

- mreadvar *assocID assocID* [*variable-name*[=*value*] [,...]]

  Similar to the readvar command except that the query is done for each of a range of (nonzero) association IDs. This range is determined from the association list cached by the most recent associations command.

- opeers

  An old form of the peers command, with the reference ID replaced by the local interface address.

- passociations

  Displays association data concerning recognized peers from the internally cached list of associations. This command performs identically to the associations command except that it displays the internally stored data rather than make a new query.

- peers

  Displays a list of recognized peers of the server, along with a summary of each peer's state. Summary information includes the address of the remote peer; the reference ID (0.0.0.0 if the reference ID is unknown); the stratum of the remote peer; the polling interval (in seconds); the reachability register (in octal); and the current estimated delay, offset, and dispersion of the peer (in milliseconds).

  The character in the left margin indicates the fate of this peer in the clock selection process. The codes are as follows:

  Space indicates that the peer was discarded, because of high stratum or failed sanity checks.
  Lowercase x indicates that the peer was designated a falseticker by the intersection algorithm.
  Dot (.) indicates that this peer was culled from the end of the candidate list.
  Hyphen (-) indicates that the peer was discarded by the clustering algorithm.
  Plus sign (+) indicates that the peer was included in the final selection set.
  Pound sign (#) indicates that the peer was selected for synchronization, but the distance exceeds the maximum.
  Asterisk (*) indicates that the peer was selected for synchronization.

Because the peers command depends on the ability to parse the values in the responses it gets, it might fail to work with servers that control the data formats poorly.

The contents of the host field can be in one of four forms: a host name, an IP address, a reference clock implementation name with its parameter, or REFCLK (implementation number parameter). If you specified hostnames no, the IP addresses will be displayed.

- pstatus *assocID*

  Sends a read status request to the server for the given association. The names and values of the peer variables returned are printed. The status word from the header is displayed preceding the variables, both in hexadecimal and in English.

- readlist [*assocID*]

  Requests that the server return the values of the variables in the internal variable list. If the association ID is omitted or is zero, the variables are assumed to be system variables. Otherwise, they are treated as peer variables. If the internal variable list is empty, a request is sent without data; the remote server should return a default display.

- readvar [*assocID*] [*variable-name*[=*value*] [,...]]

  Requests that the values of the specified variables be returned by the server by sending a read variables request. If the association ID is omitted or is zero, the variables are system variables; otherwise, they are peer variables, and the values returned are those of the corresponding peer. If the variable list is empty, a request is sent without data; the remote server should return a default display.

- showvars

  Displays the variables on the variable list.

- version

  Displays the NTPQ version number.

- writelist [*assocID*]

  Like the readlist request except that the internal list variables are written instead of read.

- writevar *assocID variable-name*=*value* [,...]

  Like the readvar request except that the specified variables are written instead of read.

Use the following syntax to enter the NTPQ foreign command:

```
NTPQ [-i] [-n] [-p] [-c command] [host1,host2,...]
```

Table B–7 describes the NTPQ options.

Table B–7 NTPQ Options

Option	Description
`-c` *command*	Adds the specified interactive command to the list of commands to be executed on the specified host. You can enter multiple `-c` options on the command line.
`-i`	Forces NTPQ to operate in interactive mode. This is the default mode of operation.
`-n`	Displays host addresses numeric format rather than converting them to host names.
`-p`	Displays a list of the peers known to the server as well as a summary of their state.

The `-c` and `-p` options send the query to the specified host immediately. If you omit the host names, the default is the local host. To enter interactive mode, specify the `-i` or `-n` option.

B.7.5 Generating Random Keys with NTP_GENKEYS

The NTP_GENKEYS program allows you to generate random keys used by the NTP Version 3 and NTP Version 4 symmetric key authentication scheme. By default, the program generates the TCPIP$NTP.KEYS file containing 16 random symmetric keys. A timestamp in NTP seconds is appended to the filename. Because the algorithms are seeded by the system clock, each run of the program produces a different file and filename.

The TCPIP$NTP.KEYS file contains 16 MD5 keys. Each key consists of 16 characters randomized over the ASCII 95-character printing subset. The file is read by the NTP server at the location specified by the `keys` configuration file command. An additional key consisting of an easily remembered password should be added by hand for use with the NTPQ and NTPDC programs. The file must be distributed by secure means to other servers and clients that share the same security compartment. The key identifier for the MD5 program uses only the identifiers from 1 to 16. The key identifier for each association is specified as the key argument in the `server` or `peer` configuration file command.

B.8 Solving NTP Problems

Some common NTP problems include:

- System clock not synchronized.

  NTP cannot synchronize a clock that is off by more than 1000 seconds. To solve this problem, set the clock using NTPDATE, then restart NTP.

- NTPDATE fails to set the clock.

  This occurs if NTP is already running.

- More than one service is actively setting the system clock.

  NTP can run with other time services but must be explicitly instructed not to set the system clock. NTP can still provide synchronization to other clients even if it is not updating the system clock.

- NTP appears to be running without error, but the system clock is off by a one-, two-, three-, or four-hour interval.

  You might need to adjust the time zone differential. For more information, please consult the OpenVMS documentation set.

B.8.1 NTP Debugging Techniques

Once the configuration file has been created and edited, the next step is to verify correct operation and then fix any problems that might have resulted.

B.8.1.1 Initial Startup

The best way to verify correct operation is by using the NTPQ and NTPDC programs, either on the server itself or from another machine elsewhere in the network. The NTPQ program implements the management functions specified in the NTP specification RFC 1305, Appendix A. The NTPDC program implements additional functions not provided in the standard. Both programs can be used to inspect the state variables defined in the specification and, in the case of NTPDC, additional ones of interest. In addition, the NTPDC program can be used to selectively reconfigure and enable or disable some functions while the server is running. Problems are apparent in the server's log file. The log file should show the startup banner, some brief initialization data, and the computed precision value.

Another common problem is incorrect DNS names. Check that each DNS name used in the configuration file exists and that the address responds to the `ping` command. When the server is first started it normally polls the servers listed in the configuration file at 64-second intervals. To allow a sufficient number of samples for the NTP algorithms to discriminate reliably between correctly operating servers and possible intruders, at least four valid messages from the majority of servers and peers listed in the configuration file are required before the server can set the local clock. However, if the difference between the client time and server time is greater than the panic threshold (which defaults to 1000 seconds), the server sends a message to the server log and shuts down without setting the clock. It is necessary to set the local clock to within the panic threshold first, either manually by wristwatch and the `SET TIME` command, or by using the NTPDATE command. The panic threshold can be changed by the `tinker panic` statement.

B.8.1.2 Verifying Correct Operation

After starting the server, run the NTPQ program with the `-n` switch to avoid distractions because of name resolution problems. Use the `peer` command to display a list showing the status of configured peers and other clients trying to access the server. After operating for a few minutes, the display should look similar to the following:

```
NTPQ> peer
     remote      refid       st t when poll reach delay offset jitter
==============================================================================
-isipc6.cairn.ne .GPS1.         1 u  18  64  377  65.592 -5.891  0.044
+saicpc-isiepc2. pogo.udel.edu 2 u 241 128  370  10.477 -0.117  0.067
+uclpc.cairn.net pogo.udel.edu 2 u  37  64  177 212.111 -0.551  0.187
*pogo.udel.edu   .GPS1.         1 u  95 128  377   0.607  0.123  0.027
```

The host names or addresses shown in the `remote` column correspond to the server and peer entries listed in the configuration file; however, the DNS names might not agree if the names listed are not the canonical DNS names. The `refid` column shows the current source of synchronization; the `st` column shows the stratum; the `t` column shows the type (`u` = unicast, `m` = multicast, `l` = local, `-` = don't know); and the `poll` column shows the poll interval in seconds. The `when` column shows the time (in seconds) since the peer was last heard, and the `reach` column shows the status of the reachability register (in octal) (see RFC 1305). The remaining entries show the latest delay, offset, and jitter (in milliseconds).

Note that in NTP Version 4 what used to be the dispersion column has been replaced by the jitter column.

The symbol at the left margin displays the synchronization status of each peer. The currently selected peer is marked with an asterisk (*), while additional peers that are not currently selected but are designated acceptable for synchronization are marked with a plus sign (+). Peers marked with * and + are included in the weighted average computation to set the local clock; the data produced by peers marked with other symbols are discarded.

Additional details for each peer can be determined by the following procedure. First, use the *associations* command to display an index of association identifiers, as shown in the following example:

```
NTPQ> associations
ind assID status  conf reach auth condition  last_event cnt
===========================================================
  1 50252  f314   yes   yes   ok    outlyer   reachable  1
  2 50253  f414   yes   yes   ok   candidat   reachable  1
  3 50254  f414   yes   yes   ok   candidat   reachable  1
  4 50255  f614   yes   yes   ok   sys.peer   reachable  1
```

Each line in this display is associated with the corresponding line in the preceding `peer` display. The `assID` column shows the unique identifier for each mobilized association, and the `status` column shows the peer status word (in hexadecimal), as defined in the NTP specification.

Next, use the `readvar` command and the respective `assID` identifier to display a detailed synopsis for the selected peer, as shown in the following example:

```
NTPQ> readvar 50253
status=f414 reach, conf, auth, sel_candidat, 1 event, event_reach,
srcadr=saicpc-isiepc2.cairn.net, srcport=123, dstadr=140.173.1.46,
dstport=123, keyid=3816249004, stratum=2, precision=-27,
rootdelay=10.925, rootdispersion=12.848, refid=pogo.udel.edu,
reftime=bd11b225.133e1437  Sat, Jul  8 2000 13:59:01.075, delay=10.550,
offset=-1.357, jitter=0.074, dispersion=1.444, reach=377, valid=7,
hmode=1, pmode=1, hpoll=6, ppoll=7, leap=00, flash=00 ok,
org=bd11b23c.01385836  Sat, Jul  8 2000 13:59:24.004,
rec=bd11b23c.02dc8fb8  Sat, Jul  8 2000 13:59:24.011,
xmt=bd11b21a.ac34c1a8  Sat, Jul  8 2000 13:58:50.672,
filtdelay=   10.45  10.50  10.63  10.40  10.48  10.43  10.49  11.26,
filtoffset=  -1.18  -1.26  -1.26  -1.35  -1.35  -1.42  -1.54  -1.81,
filtdisp=     0.51   1.47   2.46   3.45   4.40   5.34   6.33   7.28,
```

A detailed explanation of the fields in this display are beyond the scope of this manual; however, most variables defined in the NTP Version 3 specification RFC 1305 are available, along with others defined for NTP Version 4. This example was chosen to illustrate one of the most complex configurations involving symmetric modes. As the result of debugging experience, the names and values of these variables might change from time to time.

A useful indicator of miscellaneous problems is the `flash` value, which reveals the state of the various sanity tests on incoming packets. There are currently eleven bits, one for each test, numbered from right to left, which is for test 1. If the test fails, the corresponding bit is set to 1 and 0. If any bit is set following each processing step, the packet is discarded.

The three lines identified as filtdelay, filtoffset, and filtdisp reveal the round-trip delay, clock offset and dispersion for each of the last eight measurement rounds (all in milliseconds). Note that the dispersion, which is an estimate of the error, increases as the age of the sample increases. From these data, it is usually possible to determine the incidence of severe packet loss, network congestion, and unstable local clock oscillators. Every case is unique; however, if one or more of the rounds show large values or change radically from one round to another, the network is probably congested or experiencing loss.

Once the server has set the local clock, it continuously tracks the discrepancy between local time and NTP time and adjusts the local clock accordingly. This adjustment consists of two components: time and frequency. Adjustments to time and frequency are determined automatically by the clock discipline algorithm, which functions as a hybrid phase/frequency feedback loop. The behavior of this algorithm is controlled carefully to minimize residual errors resulting from normal network jitter and frequency variations of the local clock hardware oscillator. However, when started for the first time, the algorithm may take some time to converge on the intrinsic frequency error of the host machine.

The state of the local clock itself can be determined using the readvar statement (without the argument), as shown in the following example:

```
NTPQ> readvar
status=0644 leap_none, sync_ntp, 4 events, event_peer/strat_chg,
version="ntpd 4.0.99j4-r Fri Jul  7 23:38:17 GMT 2002 (1)",
processor="i386", system="FreeBSD3.4-RELEASE", leap=00, stratum=2,
precision=-27, rootdelay=0.552, rootdispersion=12.532, peer=50255,
refid=pogo.udel.edu,
reftime=bd11b220.ac89f40a  Sat, Jul  8 2002 13:58:56.673, poll=6,
clock=bd11b225.ee201472  Sat, Jul  8 2002 13:59:01.930, state=4,
phase=0.179, frequency=44.298, jitter=0.022, stability=0.001,
hostname="barnstable.udel.edu", publickey=3171372095,
params=3171372095,
refresh=3172016539
```

An explanation of most of these variables is in the RFC 1305 specification. The most useful variables are clock, which shows when the clock was last adjusted, and reftime, which shows when the server clock of refid was last adjusted. The mean millisecond time offset (phase) and deviation (jitter) monitor the clock quality, and the mean PPM frequency offset (frequency) and deviation (stability) monitor the clock stability and serve as useful diagnostic tools. NTP operators have found that these data represent useful environment and hardware alarms. If the motherboard fan or some hardware bit malfunctions, the system clock is usually the first to reflect these problems.

When nothing seems to happen in the peer display after several minutes, it might indicate a network problem. One common network problem is an access-controlled router on the path to the selected peer, or an access-controlled server using methods described in the Access Control Options section. Another common problem is that the server is down or is running in unsynchronized mode because of a local problem. Use the NTPQ program to look at the server variables in the same way you look at your own.

B.8.1.3 Special Problems

The frequency tolerance of computer clock oscillators can vary widely, which can put a strain on the server's ability to compensate for the intrinsic frequency error. While the server can handle frequency errors up to 500 parts per million (ppm), or 43 seconds per day, values much higher than 100 ppm reduce the headroom and increase the time to learn the particular value and record it in the TCPIP$NTP.DRIFT file. In extreme cases, before the particular oscillator frequency error has been determined, the residual system time offsets can sweep from one extreme to the other of the 128-millisecond tracking window only for the behavior to repeat at 900-second intervals until the measurements have converged.

To determine whether excessive frequency error is occurring, observe the nominal `filtoffset` values for a number of rounds and divide by the poll interval. If the result is approximately 500 ppm, NTP probably will not work properly until the frequency error is reduced.

A common cause of this problem is the hardware time-of-year (TOY) clock chip, which must be disabled when NTP disciplines the software clock.

If the TOY chip is not the cause, the problem might be that the hardware clock frequency is too slow or too fast.

NTPD provides for access controls that deflect unwanted traffic from selected hosts or networks. The controls described in the Access Control Options section include detailed packet filter operations based on source address and address mask. Normally, filtered packets are dropped without notice other than to increment tally counters. However, the server can be configured to generate a kiss-of-death (kod) packet to be sent to the client. If outright access is denied, the kod is the response to the first client packet. In this case, the client association is permanently disabled and the access-denied bit is set in the `flash` peer variable, and a message is sent to the server's log file.

The access control provisions include a limit on the packet rate from a host or network. If an incoming packet exceeds the limit, it is dropped and a kod is sent to the source. If this occurs after the client association has synchronized, the association is not disabled, but a message is sent to the system log. For more information, see the Access Control Options section in this chapter.

B.8.1.4 Debugging Checklist

If the NTPQ or NTPDC programs do not show that messages are being received by the server or that received messages do not result in correct synchronization, verify the following:

1. Check the TCPIP$NTP_RUN.LOG log file for messages about configuration errors, name-lookup failures, or initialization problems.
2. Using `ping` or other utilities,verify that packets actually do make the round trip between the client and server. Using `dig` or other utilities, verify that the DNS server names do exist and resolve to valid Internet addresses.
3. Using the NTPDC program, verify that the packets received and packets sent counters are incrementing. If the sent counter does not increment and the configuration file includes configured servers, something might be wrong in the host network or the interface configuration. If this counter does increment but the received counter does not increment, something might be wrong in the

network, the remote server NTP server might not be running, or the server itself might be down or not responding.

4. If both the sent and received counters do increment but the `reach` values in the `peer` display with NTPQ continues to show zero, received packets are probably being discarded. If this is the case, the cause should be evident from the `flash` variable.

5. If the `reach` values in the `peer` display show that the servers are alive and responding, note the symbols at the left margin that indicate the status of each server resulting from the various grooming and mitigation algorithms. After a few minutes of operation, one of the reachable server candidates should show an asterisk (*) symbol. If this does not happen, the intersection algorithm, which classifies the servers as truechimers or falsetickers, might be unable to find a majority of truechimers among the server population.

C

Configuring and Managing BIND Version 9

The Domain Name System (DNS) maintains and distributes information about Internet hosts. DNS consists of a heirarchical database containing the names of entities on the Internet, the rules for delegating authority over names, and mail routing information; and the system implementation that maps the names to Internet addresses.

In OpenVMS environments, DNS is implemented by the Berkeley Internet Name Domain (BIND) software. Compaq TCP/IP Services for OpenVMS implements a BIND server based on the Internet Software Consortium's (ISC) BIND Version 9.

This chapter contains the following topics:

- How to migrate your existing BIND 4 environment to BIND 9 (Section C.3)
- How to configure BIND using the BIND configuration file (Section C.5), including:
  - How to configure dynamic updates (Section C.5.7)
  - How to configure a DNS cluster failover and redundancy environment (Section C.5.8)
- How to populate the BIND server databases (Section C.6)
- How to examine name server statistics (Section C.7)
- How to configure BIND using SET CONFIGURATION BIND commands (Section C.8)
- How to configure the BIND resolver (Section C.9)
- How to use the BIND server administrative tools (Section C.10)
- How to troubleshoot BIND server problems (Section C.11)

C.1 Key Concepts

This section serves as a review only and assumes you are acquainted with the InterNIC, that you applied for an IP address, and that you registered your domain name. You should also be familiar with BIND terminology, and you should have completed your preconfiguration planning before using this chapter to configure and manage the BIND software.

If you are not familiar with DNS and BIND, see the *Compaq TCP/IP Services for OpenVMS Concepts and Planning* guide. If you need more in-depth knowledge, see O'Reilly's *DNS and BIND, Fourth Edition*. You can find the *BIND 9 Adminstrator Reference Manual* at `http://www.isc.org/`.

C.1.1 How the Resolver and Name Server Work Together

BIND is divided conceptually into two components: a resolver and a name server. The resolver is software that queries a name server; the name server is the software process that responds to a resolver query.

Under BIND, all computers use resolver code, but not all computers run the name server process.

The BIND name server runs as a distinct process called TCPIP$BIND. On UNIX systems, the name server is called `named` (pronounced name-dee). Name servers are typically classified as master (previously called primary), slave (previously called secondary), and caching-only servers, depending on their configurations.

C.1.2 Common BIND Configurations

You can configure BIND in several different ways. The most common configurations are resolver-only systems, master servers, slave servers, forwarder servers, and caching-only servers. A server can be any of these configurations or can combine elements of these configurations.

Servers use a group of database files containing BIND statements and resource records. These files include:

- The forward translation file, *domain_name*.DB

  This file maps host names to IP addresses.

- The reverse translation file, *address*.DB

  This file maps the address back to the host names. This address name lookup is called reverse mapping. Each domain has its own reverse mapping file.

- Local loopback forward and reverse translation files, LOCALHOST.DB and 127_0_0.DB

  These local host databases provide forward and reverse translation for the widely used LOCALHOST name. The LOCALHOST name is always associated with IP address 127.0.0.1 and is used for loopback traffic.

- The hint file, ROOT.HINT

  This file contains the list of root name servers.

A configuration file, TCPIP$BIND.CONF, contains statements that pull all the database files together and governs the behavior of the BIND server.

C.1.2.1 Master Servers

A master server is the server from which all data about a domain is derived. Master servers are **authoritative**, which means they have complete information about their domain and that their responses are always accurate.

To provide central control of host name information, the master server loads the domain's information directly from a disk file created by the domain administrator. When a new system is added to the network, only the database on the master server needs to be modified.

A master server requires a complete set of configuration files: zone, reverse domain, configuration, hint, and loopback files.

C.1.2.2 Slave Servers

Slave servers receive authority and their database from the master server.

A particular domain's database file is called a **zone** file; copying this file to a slave server is called a **zone file transfer**. A slave server assures that it has current information about a domain by periodically transferring the domain's zone file. Slave servers are also authoritative for their domain.

Configuring a slave server is similar to configuring a master server. The only difference is that, for the slave server, you need to provide the name of the master server from which to transfer zone data.

Note

If you create a master, slave, or forwarder server for the same domain on which your local host resides, you should reconfigure your BIND resolver so that it uses this system (LOCALHOST) as its name server.

Slave servers require a configuration file, a hint file, and loopback files.

C.1.2.3 Caching-Only Servers

Caching-only servers get the answers to all name service queries from other name servers. Once a caching server receives an answer to a query, it saves the information and uses it in the future to answer queries itself. Most name servers cache answers and use them in this way but a caching-only server depends on this for all its server information. It does not keep name server database files as other servers do. Caching-only servers are **nonauthoritative**, which means that their information is secondhand and can be incomplete.

Caching-only servers require a hint file and loopback files.

C.1.2.4 Forwarder Servers

The forwarding facility can be used to create a large, sitewide cache on a few servers, thereby reducing traffic over links to external name servers. Forwarder servers process requests that slave servers cannot resolve locally (for example, because they do not have access to the Internet).

Forwarding occurs on only those queries for which the server is not authoritative and for which it does not have the answer in its cache.

A master or slave server specifies a particular host to which requests outside the local zone are sent. This is a form of Internet courtesy that limits the number of hosts that actually communicate with the root servers listed in the ROOT.HINT file.

If you configure a forwarder server, you must provide the name of the host to which requests outside your zones of authority are forwarded.

C.2 Security Considerations

BIND Version 9 provides the following security enhancements:

- Access control lists allow you to control access to the name server. See Section C.2.1 for more information.
- Dynamic Update Security controls access to the dynamic update facility. See Section C.2.2 for more information.

- Transaction Signatures (TSIG) provide key-based access to the dynamic update facility. See Section C.2.3 for more information.
- TKEY automatically generates a shared secret between two hosts. See Section C.2.4 for more information.
- SIG(0) is another method for signing transactions. See Section C.2.5 for more information.
- DNSSEC provides cryptographic authentication of DNS information. See Section C.2.6 for more information.

C.2.1 Access Control Lists

Access control lists (ACLs) are address match lists that you can set up and name for use in configuring the following options:

- `allow-notify`
- `allow-query`
- `allow-recursion`
- `blackhole`
- `allow-transfer`

Using ACLs, you can control who can access your name server without cluttering your configuration files with huge lists of IP addresses.

It is a good idea to use ACLs and to control access to your server. Limiting access to your server by outside parties can help prevent unwanted use of your server.

Here is an example of how to apply ACLs properly:

```
      // Set up an ACL named "bogusnets" that will block RFC1918 space,
      // which is commonly used in spoofing attacks.
acl bogusnets { 0.0.0.0/8; 1.0.0.0/8; 2.0.0.0/8; 192.0.2.0/24;
  224.0.0.0/3; 10.0.0.0/8; 172.16.0.0/12; 192.168.0.0/16;
};
      // Set up an ACL called our-nets. Replace this with the real IP numbers.
acl our-nets { x.x.x.x/24; x.x.x.x/21; };
options {
  ...
  ...
  allow-query { our-nets; };
  allow-recursion { our-nets; };
  ...
  blackhole { bogusnets; };
  ...
};
zone "example.com" {
  type master;
  file "example_com.db";
  allow-query { any; };
};
```

This example allows recursive queries of the server from the outside, unless recursion has been previously disabled. For more information about how to use ACLs to protect your server, see Section C.5.2.

C.2.2 Dynamic Update Security

Access to the dynamic update facility should be strictly limited. In earlier versions of BIND, the only way to do this was to include an IP address or network prefix in the `allow-update` zone option. This method is insecure because the source address of the update UDP packet is easily forged. Also, if the IP addresses allowed by the `allow-update` option include the address of a slave server that performs forwarding of dynamic updates, the master can be trivially attacked by sending the update to the slave, which will forward it to the master with its own source IP address. This causes the master to approve the update without question.

For these reasons, updates should be authenticated cryptographically by means of transaction signatures (TSIG). That is, the `allow-update` option should list only TSIG key names, not IP addresses or network prefixes. Alternatively, you can use the new `update-policy` option.

Some sites choose to keep all dynamically updated DNS data in a subdomain and to delegate that subdomain to a separate zone. This way, the top-level zone containing critical data, such as the IP addresses of public web and mail servers, need not allow dynamic updates at all.

For information about setting up dynamic updates, see Section C.5.7.

C.2.3 TSIG

This section describes how to set up Transaction Signatures (TSIG) transaction security in BIND. It describes changes to the configuration file as well as the changes that are required for different features, including the process of creating transaction keys and how to use transaction signatures with BIND.

BIND primarily supports TSIG for server-to-server communication. This includes zone transfer, notify, and recursive query messages.

TSIG is useful for dynamic updating. A primary server for a dynamic zone should use access control to control updates, but IP-based access control is insufficient. Key-based access control is far superior. To use TSIG with the `nsupdate` utility, specify either the `-k` or `-y` option on the NSUPDATE command line. For more information about using the `nsupdate` utility, see Section C.5.7.3.

Use the following procedure to implement TSIG:

1. Generate shared keys for each pair of hosts.

   You can generate shared keys automatically, or you can specify them manually. In the example that follows, a shared secret is generated to be shared between HOST1 and HOST2. The key name is `host1-host2`. The key name must be the same on both hosts.

   Longer keys are better, but shorter keys are easier to read. The maximum key length is 512 bits; keys longer than that will be digested with MD5 to produce a 128-bit key. Use the `dnssec-keygen` utility to generate keys automatically.

   The following command generates a 128-bit (16-byte) HMAC-MD5 key:

   ```
   $ dnssec_keygen -a hmac-md5 -b 128 -n HOST host1-host2.
   ```

In this example, the key is in the file KHOST1-HOST2.157-00000_PRIVATE. Nothing uses this file directly, but the base-64 encoded string following Key: can be extracted from the file and can be used as a shared secret. For example:

```
Key: La/E5CjG9O+os1jq0a2jdA==
```

The string La/E5CjG9O+os1jq0a2jdA== can be used as the shared secret.

Keys can also be specified manually. The shared secret is a random sequence of bits, encoded in base-64. Most ASCII strings are valid base-64 strings (assuming the length is a multiple of 4 and that only valid characters are used).

2. Copy the shared secret to both hosts.

   Use a secure transport mechanism like a floppy disk, or a physically secure network, to copy the shared secret between hosts.

3. Inform the servers of the key's existence.

   In the following example, HOST1 and HOST2 are both servers. Add the following to each server's TCPIP$BIND.CONF file:

   ```
   key host1-host2. {
     algorithm hmac-md5;
     secret "La/E5CjG9O+os1jq0a2jdA==";
   };
   ```

   The HMAC-MD5 algorithm is the only one supported by BIND. It is recommended that either TCPIP$BIND.CONF not be world readable, or that the key statement be added to a nonworld readable file that is included by TCPIP$BIND.CONF. For information about the key statement, see Section C.5.3.4.

   Once the configuration file is reloaded, the key is recognized. This means that if the server receives a message signed by this key, it can verify the signature. If the signature succeeds, the response is signed by the same key.

4. Instruct the server to use the key.

   Because keys are shared only between two hosts, the server must be told when keys are to be used. Add the following to the TCPIP$BIND.CONF file for HOST1. The IP address of HOST2 is 10.1.2.3.

   ```
   server 10.1.2.3 {
     keys { host1-host2. ;};
   };
   ```

   Multiple keys can be present, but only the first is used. This statement does not contain any secrets, so you can include it in a world-readable file.

   If HOST1 sends a message that is a request to that address, the message will be signed with the specified key. HOST1 will expect any responses to signed messages to be signed with the same key.

   A similar statement must be present in HOST2's configuration file (with HOST1's address) for HOST2 to sign request messages to HOST1.

5. Implement TSIG key-based access control.

   You can specify TSIG keys in ACL definitions and in the following configuration options:

   - allow-query
   - allow-transfer

- allow-update

For the key named HOST1-HOST2., specify the following allow-update option:

```
allow-update { key host1-host2. ;};
```

This statement allows dynamic updates to succeed only if the request was signed by a key named HOST1-HOST2.

6. Reload the configuration file.

   Changes to the configuration file will not take effect until the configuration file is reloaded. You can use one of several methods to reload the configuration file:

   - The rndc utility
   - The TCP/IP management command SET NAME/INITIALIZE
   - Stopping and restarting the BIND server

7. Handle any errors.

   The processing of TSIG-signed messages can result in several types of errors. If a signed message is sent to a non-TSIG aware server, an error is returned because the server will not understand the record. This is a result of misconfiguration; the server must be configured explicitly to send a TSIG-signed message to a specific server.

   If a TSIG-aware server receives a message signed by an unknown key, the response is unsigned and an error is returned.

   If a TSIG-aware server receives a message with a signature that is not validated, the response is unsigned and an error is returned.

   If a TSIG aware server receives a message with a time outside of the allowed range, the response is signed, an error is returned, and the time values are adjusted so that the response can be successfully verified.

C.2.4 TKEY

TKEY is a mechanism for automatically generating a shared secret between two hosts. There are several modes of TKEY that specify how the key is generated or assigned. BIND implements only the Diffie-Hellman key exchange. Both hosts are required to have a Diffie-Hellman KEY record (although this record is not required to be present in a zone). The TKEY process must use messages signed either by TSIG or SIG(0). The result of TKEY is a shared secret that can be used to sign messages with TSIG. TKEY can also be used to delete shared secrets that it had previously generated.

The TKEY process is initiated by a client or server by sending a signed TKEY query (including any appropriate KEYs) to a TKEY-aware server. The server response, if it indicates success, contains a TKEY record and any appropriate keys. After this exchange, both participants have enough information to determine the shared secret. When Diffie-Hellman keys are exchanged, the shared secret is derived by both participants.

C.2.5 SIG(0)

BIND Version 9 partially supports DNSSEC SIG(0) transaction signatures (as specified in RFC 2535).

SIG(0) uses public and private keys to authenticate messages. Access control is performed in the same manner as TSIG keys; privileges can be granted or denied based on the key name. When a SIG(0) signed message is received, it is verified only if the key is known and trusted by the server; the server does not attempt to locate and validate the key.

SIG(0) signing of multiple-message TCP streams is not supported. BIND Version 9 does not include any tools that generate SIG(0) signed messages.

C.2.6 DNSSEC

Cryptographic authentication of DNS information is implemented using the DNS Security (DNSSEC) extensions (defined in RFC 2535). This section describes how to create and use DNSSEC signed zones.

BIND Version 9 provides several tools that are used in the process of creating and using DNSSEC signed zones. These tools include:

- The `dnssec_keygen` utility, which generates keys for DNSSEC (secure DNS) and TSIG (transaction signatures).
- The `dnssec_makekeyset` utility, which generates a key set.
- The `dnssec_signkey` utility, which signs a key set.
- The `dnssec_signzone` utility, which signs a zone.

For detailed information about these utilities, see Section C.10. In all cases, the `-h` option displays a full list of parameters. Note that the DNSSEC tools require the `keyset` and `signedkey` files to be in the working directory.

Administrators of the parent and child zones must agree on keys and signatures. A zone's security status must be indicated by the parent zone for a DNSSEC-capable resolver to trust its data.

For other servers to trust data in this zone, they must be statically configured either with this zone's zone key or with the zone key of another zone above this one in the DNS tree.

Use the following procedure to set up DNSSEC secure zones:

1. Generate keys.

   To generate keys, use the `dnssec_keygen` program.

   A secure zone must contain one or more zone keys. The zone keys sign all other records in the zone, as well as the zone keys of any secure delegated zones. Zone keys must have the same name as the zone, must have a name type of ZONE, and must be usable for authentication.

   The following command generates a 768-bit DSA key for the `child.example` zone:

   ```
   $ dnssec_keygen -a DSA -b 768 -n ZONE child.example.
   ```

   Two output files are produced: KCHILD_EXAMPLE.003-12345_KEY and KCHILD_EXAMPLE.003-12345_PRIVATE (where 12345 is the key tag). The key file names contain the key name (`child_example.`), the algorithm (3 is DSA, 1 is RSA), and the key tag (12345, in this case). The private key (in

the _PRIVATE file) is used to generate signatures, and the public key (in the _KEY file) is used verify signatures.

To generate another key with the same properties (but with a different key tag), repeat the preceding command.

Insert the public keys into the zone file using $INCLUDE statements that specify the _KEY files.

2. Create a key set.

   To create a key set from one or more keys, use the dnssec_makekeyset utility.

   After the zone keys are generated, a key set must be built for transmission to the administrator of the parent zone, so that the parent zone can sign the keys with its own zone key and indicate correctly the security status of this zone. When building a key set, the list of keys to be included and the TTL value of the set must be specified; the desired signature validity period of the parent's signature can also be specified.

   The list of keys to be inserted into the key set can also include nonzone keys present at the top of the zone.

   The following command generates a key set containing the key generated in the preceding example and another key similarly generated, with a TTL of 3600 and a signature validity period of 10 days starting from now:

```
$ dnssec_makekeyset -t 3600 -e +864000 KCHILD_EXAMPLE.003-12345 -
_$ KCHILD_EXAMPLE.003-23456
```

   One output file is produced: KEYSET-CHILD_EXAMPLE.DAT. This file should be transmitted to the parent to be signed. It includes the keys as well as signatures covering keys. The signatures prove that you have the private key that corresponds to the public key. The signatures also have the validity period encoded in them.

3. Sign the child's keyset.

   To sign one child's keyset, use the dnssec_signkey utility.

   If the child.example zone has any delegations which are secure (for example, grand.child.example), the child.example administrator should receive keyset files for each secure subzone. These keys must be signed by this zone's zone keys.

   The following command signs the child's key set with the zone keys:

```
$ dnssec_signkey KEYSET-GRAND_CHILD_EXAMPLE.DAT KCHILD_EXAMPLE.003-12345 -
_$ KCHILD_EXAMPLE.003-23456
```

   One output file is produced: SIGNEDKEY-GRAND_CHILD_EXAMPLE.DAT. This file should be both transmitted back to the child and retained. It includes all keys (the child's keys) from the keyset file and signatures generated by this zone's zone keys.

4. Sign the zone.

   To sign a zone, use the dnssec_signzone utility.

   Any signed key files corresponding to secure subzones should be present, as well as a signed key file for this zone generated by the parent (if there is one). The zone signer generates NXT and SIG records for the zone, incorporates the zone key signature from the parent, and indicates the security status at all delegation points.

Before signing the zone, add the KEY record to the zone database file using the $INCLUDE statement. For example:

```
$INCLUDE KCHILD_EXAMPLE.003-12345_KEY
```

The following command signs the zone, assuming it is in a file called ZONE_CHILD_EXAMPLE.DB. By default, all zone keys that have an available private key are used to generate signatures.

```
$ dnssec_signzone -o child.example ZONE_CHILD_EXAMPLE.DB
```

One output file is produced: ZONE_CHILD_EXAMPLE.DB_SIGNED. This file should be referenced by TCPIP$BIND.CONF as the input file for the zone.

5. Configure the servers.

   Unlike BIND Version 8, data is not verified when the BIND Version 9 software is loaded. Therefore, zone keys for authoritative zones do not need to be specified in the configuration file. The public key for any security root must be present in the configuration file's `trusted-keys` statement, as described in Section C.5.

C.3 Migrating from BIND Version 4 to BIND Version 9

If you set up your BIND environment using an old version of the TCP/IP Services product, you must convert the UCX databases and configuration information to the BIND Version 9 format.

To convert your BIND configuration, enter the following command:

```
TCPIP> CONVERT/CONFIGURATION BIND
```

This command extracts the BIND-specific configuration information from UCX$CONFIGURATION.DAT and creates the BIND Version 9 configuration file TCPIP$BIND.CONF. It renames your BIND databases, where necessary.

You can continue to use the SET CONFIGURATION BIND commands to make changes to your configuration (see Section C.8), or you can make changes by editing the text file TCPIP$BIND.CONF (see Section C.5). If you continue to use the SET CONFIGURATION BIND commands, you must also enter the CONVERT/CONFIGURATION BIND command in order for your changes to take effect.

C.3.1 Navigating Two Different BIND Environments

This section summarizes the differences between the UCX BIND implementation and BIND Version 9.

Remember that, in BIND Version 9, name servers are configured by editing a text configuration file. The use of this file is described in Section C.5. Compaq recommends, but does not require, that you use the configuration file to set up BIND. You can continue using the TCPIP$CONFIG and the SET CONFIGURATION BIND commands to set up your BIND environment, as you did with previous releases of this product. The term UCX BIND in Table C–1 describes the previous configuration method even though this method is still valid in the current release.

Table C–1 describes changes to the database and configuration file names.

Table C–1 UCX BIND and BIND Version 9 Differences

Database/File Names	UCX BIND	BIND Version 9
Configuration information	UCX$CONFIGURATION.DAT	TCPIP$BIND.CONF
Local loopback files	NAMED.LOCAL	LOCALHOST.DB, 127_0_0.DB
Forward lookup file	*domain_name*.DB	*domain_name*.DB
Reverse lookup file	*address*.DB	*address*.DB
Cache file	NAMED.CA	ROOT.HINT

Note

You must be consistent when making changes to your BIND environment. If you make changes by editing the configuration file, you should continue to make changes in that manner.

If you revert to the UCX BIND configuration method (SET CONFIGURATION BIND and CONVERT/CONFIGURATION BIND commands), any changes you made to the configuration file (TCPIP$BIND.CONF) are lost.

If you continue to use the SET CONFIGURATION BIND commands, you must always enter the CONVERT/CONFIGURATION BIND command in order for your changes to take effect.

C.4 BIND Service Startup and Shutdown

The BIND service can be shut down and started independently of TCP/IP Services. The following files are provided for this purpose:

- SYS$STARTUP:TCPIP$BIND_STARTUP.COM allows you to start the BIND service.
- SYS$STARTUP:TCPIP$BIND_SHUTDOWN.COM allows you to shut down the BIND service.

To preserve site-specific parameter settings and commands, create the following files. These files are not overwritten when you reinstall TCP/IP Services.

- SYS$STARTUP:TCPIP$BIND_SYSTARTUP.COM can be used as a repository for site-specific definitions and parameters to be invoked when the BIND service is started.
- SYS$STARTUP:TCPIP$BIND_SYSHUTDOWN.COM can be used as a repository for site-specific definitions and parameters to be invoked when the BIND service is shut down.

C.5 Configuring the BIND Server

This section describes how to configure the BIND name server on your local host.

BIND Version 9 configuration is broadly similar to BIND Version 8; however, there are a few new areas of configuration, such as views. BIND Version 8 configuration files should work with few alterations in BIND Version 9, although you should review more complex configurations to see whether they can be implemented more efficiently using the new features in BIND Version 9.

BIND Version 9 stores configuration information in a text file called TCPIP$BIND.CONF. The TCP/IP Services product provides a template for this file located in the SYS$SPECIFIC:[TCPIP$BIND] directory. Edit this template to reflect your site-specific configuration requirements before running BIND.

C.5.1 Configuration File Elements

Table C–2 lists the elements used throughout the BIND Version 9 configuration file documentation.

Table C–2 Name Server Configuration File Elements

Element	Description
`acl_name`	The name of an address_match_list as defined by the `acl` statement.
`address_match_list`	A list of one or more of the following elements: • `ip_addr` • `ip_prefix` • `key_id` • `acl_name` See Section C.5.2 for more information.
`domain_name`	A quoted string that will be used as a DNS name. For example: `"my.test.domain"`
`dotted_decimal`	One or more integers valued 0 through 255 and separated by dots, such as 123, 45.67 or 89.123.45.67.
`ip4_addr`	An IPv4 address with exactly four elements in dotted decimal notation.
`ip6_addr`	An IPv6 address, such as `fe80::200:f8ff:fe01:9742`.
`ip_addr`	An `ip4_addr` or `ip6_addr`.
`ip_port`	An IP port number from 0 to 65535. Values below 1024 are restricted to well-known processes. In some cases, an asterisk (*) character can be used as a placeholder to select a random high-numbered port.

(continued on next page)

Table C–2 (Cont.) Name Server Configuration File Elements

Element	Description
ip_prefix	An IP network specified as an ip_addr, followed by a slash (/) and then the number of bits in the netmask. Trailing zeros in an ip_addr can be omitted. For example, 127/8 is the network 127.0.0.0 with netmask 255.0.0.0 and 1.2.3.0 /28 is network 1.2.3.0 with netmask 255.255.255.240.
key_id	A domain name representing the name of a shared key, to be used for transaction security.
key_list	A list of one or more key_ids, separated by semicolons and ending with a semicolon.
number	A nonnegative integer with an entire range limited by the range of a C language signed integer (2,147,483,647 on a machine with 32-bit integers). Its acceptable value might be limited further by the context in which it is used.
path_name	A quoted string that will be used as a path name. For example: "SYS$SPECIFIC:[TCPIP$BIND]"
size_spec	A number, the word unlimited, or the word default. The maximum value of size_spec is that of unsigned long integers on the machine. An unlimited size_spec requests unlimited use, or the maximum available amount. A default size_spec uses the limit that was in force when the server was started. A number can optionally be followed by a scaling factor: • K (or k) for kilobytes, which scales by 1024 • M (or m) for megabytes, which scales by 1024*1024 • G (or g) for gigabytes, which scales by 1024*1024*1024 Integer storage overflow is silently ignored during conversion of scaled values, resulting in values less than intended, possibly even negative. Using the unlimited keyword is the best way to safely set a really large number.
yes_or_no	Either yes or no. The words true and false are also accepted, as are the numbers 1 and 0.
dialup_option	One of the following: • yes • no • notify • notify-passive • refresh • passive When used in a zone, notify-passive, refresh, and passive are restricted to slave and stub zones.

C.5.2 Address Match Lists

Address match lists are used primarily to determine access control for server operations. They are also used to define priorities for querying other name servers and to set the addresses on which the BIND server listens for queries. The following example shows the syntax of the address match list:

```
address_match_list = address_match_list_element ;
  [ address_match_list_element; ... ]
address_match_list_element = [ ! ] (ip_address [/length] |
   key key_id | acl_name | { address_match_list } )
```

The elements that constitute an address match list can be any of the following:

- An IP address (IPv4 or IPv6)
- An IP prefix (in the / notation)
- A key ID, as defined by the `key` statement
- The name of an address match list previously defined with the `acl` statement
- A nested address match list enclosed in braces

Elements can be negated with a leading exclamation mark (!). The match list names `any`, `none`, `localhost`, and `localnets` are predefined. More information on those names can be found in the description of the `acl` statement (see Section C.5.3.1).

When a given IP address or prefix is compared to an address match list, the list is traversed in order until an element matches. The interpretation of a match depends on whether the list is being used for access control, defining listen-on ports, or as a topology, and whether the element was negated. Specifically:

- When used as an access control list, a non-negated match allows access and a negated match denies access. If there is no match, access is denied. The following options use address match lists for this purpose:
  - `allow-notify`
  - `allow-query`
  - `allow-transfer`
  - `allow-update`
  - `blackhole`

  The `listen-on` option causes the server not to accept queries on any of the machine's addresses that do not match the list.
- When used with the `topology` statement, a nonnegated match returns a distance based on its position on the list; the closer the match is to the start of the list, the shorter the distance between it and the server. A negated match is assigned the maximum distance from the server. If there is no match, the address gets a distance that is further than any nonnegated list element and closer than any negated element.

Because of the first-match aspect of the algorithm, an element that defines a subset of another element in the list should come before the broader element, regardless of whether either is negated. For example, in `1.2.3/24; ! 1.2.3.13;`, the 1.2.3.13 element is ignored, because the algorithm will match any lookup for 1.2.3.13 to the `1.2.3/24` element. Using `! 1.2.3.13; 1.2.3/24` corrects that problem by having 1.2.3.13 blocked by the negation, while all other 1.2.3.* hosts fall through.

C.5.3 Configuration File Format

A BIND configuration file consists of statements and comments. Statements end with a semicolon. Many statements contain a block of substatements that also end with a semicolon. Table C–3 describes the configuration statements.

Table C–3 BIND Name Server Configuration Statements

Statement	Description
acl	Specifies a named IP address matching list, for access control and other uses.
controls	Declares control channels to be used by the rndc utility.
include	Includes a file.
key	Specifies key information for use in authentication and authorization using TSIG. See Section C.2.3 for more information.
logging	Specifies what the server logs, and where the log messages are sent.
options	Controls global server configuration options and sets defaults for other statements.
server	Sets configuration options, and sets defaults for other statements.
trusted-keys	Specifies trusted DNSSEC keys.
view	Specifies a view.
zone	Specifies a zone.

The following sample is a configuration file for a master server:

```
options {
        directory "SYS$SPECIFIC:[TCPIP$BIND]";
};
zone "FRED.PARROT.BIRD.COM" in {
        type master;
        file "FRED_PARROT_BIRD_COM.DB";
};

zone "0.0.127.IN-ADDR.ARPA" in {
        type master;
        file "127_0_0.DB";
};
zone "LOCALHOST" in {
        type master;
        file "LOCALHOST.DB";
};
zone "208.20.16.IN-ADDR.ARPA" in {
        type master;
        file "208_20_16_IN-ADDR_ARPA.DB";
};
zone "." in {
        type hint;
        file "ROOT.HINT";
};
```

The following comment styles are valid in a BIND configuration file. Comments can appear anywhere in the file.

- C-style comments that start with /* and end with */
- C++ style comments that start with // and continue to the end of the physical line
- Shell or Perl-style comments that start with # and continue to the end of the physical line

Note

Do not use a semicolon (;) as a comment character in your configuration file. The semicolon indicates the end of a configuration statement; whatever follows is interpreted as the start of the next statement.

C.5.3.1 The ACL Statement

The `acl` statement assigns a symbolic name to an address match list. It gets its name from a primary use of address match lists: access control lists (ACLs).

Note

The access control lists used by the BIND service and OpenVMS ACLs are different structures with different purposes.

The `acl` statement is formatted as follows:

```
acl acl-name {
    address_match_list
};
```

Note that the address match list must be defined with `acl` before it can be used elsewhere; forward references are not allowed.

The following ACLs are created automatically:

ACL	Matches
`any`	All hosts
`none`	No hosts
`localhost`	The IPv4 addresses of all interfaces on the system
`localnets`	Any host on an IPv4 network for which the system has an interface

The ACLs `localhost` and `localnets` do not support IPv6. The ACL `localhost` does not match the host's IPv6 addresses, and the ACL `localnets` does not match the host's attached IPv6 networks. This limitation is due to the fact that there is no standard method for determining the complete set of local IPv6 addresses for a host.

C.5.3.2 The CONTROLS Statement

The controls statement declares control channels to be used by system administrators to affect the operation of the local name server. These control channels are used by the rndc utility to send commands to, and retrieve non-DNS results from, a name server. The controls statement is formatted as follows:

```
controls {
   inet ( ip_addr | * ) [ port ip_port ] allow {  address_match_list  }
                keys {  key_list  };
   [ inet ...; ]
};
```

An inet control channel is a TCP/IP socket accessible to the Internet, created at the specified *ip_port* on the specified *ip_addr*. If no port is specified, port 953 is used by default. The asterisk character (*) cannot be used for *ip_port*.

The ability to issue commands over the control channel is restricted by the allow and keys clauses. Connections to the control channel are permitted based on the address permissions in the address match list. key_id members of the address match list are ignored, and instead are interpreted independently based the *key_list*. Each key_id in the *key_list* is used to authenticate commands and responses given over the control channel, by digitally signing each message between the server and a command client. In order to be honored, all commands to the control channel must be signed by one of its specified keys.

If no controls statement is present, the BIND server will set up a default control channel listening on the loopback address 127.0.0.1 and its IPv6 counterpart ::1. In this case, and also when the controls statement is present but does not have a keys clause, the BIND server will attempt to load the command channel key from the file RNDC.KEY in TCPIP$ETC. To create a RNDC.KEY file, use the following command:

```
rndc_confgen -a
```

See Section C.10 for more information about using the rndc utility.

The RNDC.KEY feature eases the transition of systems from BIND Version 8, which did not have digital signatures on its command channel messages and thus did not have a keys clause. You can use an existing BIND Version 8 configuration file in BIND Version 9 unchanged, and rndc will work the same way that ndc worked in BIND Version 8.

Because the RNDC.KEY feature is only intended to allow the backward-compatible usage of BIND Version 8 configuration files, this feature does not have a high degree of configurability. You cannot easily change the key name or the size of the secret. You should make a RNDC.CONF file with your own key if you wish to change those things.

The UNIX control channel type of BIND Version 8 is not supported in BIND Version 9, and is not expected to be added in future releases. If it is present in the controls statement from a BIND Version 8 configuration file, it is ignored and a warning is logged.

C.5.3.3 The INCLUDE Statement

The include statement inserts the specified file at the point that the include statement is encountered. The include statement facilitates the administration of configuration files by permitting the reading or writing of some things but not others. For example, the statement could include private keys that are readable only by a name server. The following example shows the format of the include statement:

```
include filename;
```

C.5.3.4 The KEY Statement

The key statement defines a shared secret key for use with TSIG (see Section C.2.3). The following example shows the format of the key statement:

```
key key_id {
    algorithm algorithm-id;
    secret secret-string;
};
```

The key statement can occur at the top level of the configuration file or inside a view statement. Keys defined in top-level key statements can be used in all views. Keys intended for use in a controls statement must be defined at the top level.

Table C–4 describes the elements of the key statement.

Table C–4 Key Statement Elements

Element	Description
key_id	Specifies a domain name that uniquely identifies the key (also known as the key name). It can be used in a server statement to cause requests sent to that server to be signed with this key, or in address match lists to verify that incoming requests have been signed with a key matching this name, algorithm, and secret.
algorithm-id	Specifies an authentication algorithm. The only algorithm currently supported with TSIG authentication is HMAC-MD5.
secret-string	Specifies the secret to be used by the algorithm; treated as a Base-64 encoded string.

C.5.3.5 The LOGGING Statement

The logging statement configures a wide variety of logging options for the name server. Its channel phrase associates output methods, format options and severity levels with a name that can then be used with the category phrase to select the way each class of message is logged. The following example shows the format of the logging statement:

```
logging {
   [ channel channel_name {
     ( file path_name
       | syslog
       | stderr
       | null );
     [ severity (critical | error | warning | notice |
                 info | debug [ level ] | dynamic ); ]
     [ print-category yes_or_no; ]
     [ print-severity yes_or_no; ]
     [ print-time yes_or_no; ]
   }; ]
   [ category category_name {
     channel_name ; [ channel_name ; ... ]
   }; ]
   ...
};
```

Use one logging statement to define as many channels and categories as you want. If there is no logging statement, the logging configuration defaults to the following:

```
logging {
     category "unmatched" { "null"; };
     category "default" { "default_syslog"; "default_debug"; };
};
```

In BIND Version 9, the logging configuration is only established after the entire configuration file has been parsed. In BIND Version 8, it was established as soon as the logging statement was parsed. When the server is starting up, all logging messages regarding syntax errors in the configuration file go to the default channels.

C.5.3.5.1 The Channel Phrase All log output goes to one or more channels; you can make as many of them as you want. Every channel definition must include a destination clause that says whether messages selected for the channel go to a file (by default, TCPIP$BIND_RUN.LOG), or are discarded. It can optionally also limit the message severity level that is accepted by the channel (the default is `info`); and can specify whether to include a time stamp, the category name and severity level (the default is not to include any).

The `null` destination clause causes all messages sent to the channel to be discarded; in that case, other options for the channel are meaningless.

The `file` destination clause directs the channel to a disk file.

On OpenVMS, the `syslog` and `stderr` destination clauses direct the channel to the TCPIP$BIND_RUN.LOG file.

The `severity` clause allows you to specify the level of diagnostic messages to be logged.

The server can supply extensive debugging information when it is in debugging mode. If the server's global debug level is greater than zero, then debugging mode is activated. The global debug level is set by one of the following:

- Starting the BIND server with the `-d` flag followed by a positive integer.
- Entering the `trace` command with the `rndc` utility. To set the global debug level to zero and turn off debugging mode, enter the the `notrace` command.

All debugging messages in the server have a debug level; higher debug levels give more detailed output. Channels that specify a debug severity level get the specified debugging output and less any time the server is in debugging mode, regardless of the global debugging level. For example, to produce debugging output at level 3 and less, enter the following:

```
channel "specific_debug_level" {
    file "foo";
    severity debug 3;
};
```

Channels with dynamic severity use the server's global level to determine what messages to display.

If print-time is turned on, the date and time are logged. If print-category is requested, then the category of the message is logged as well. Finally, if print-severity is turned on, then the severity level of the message is logged. The print- options can be used in any combination and are always displayed in the following order:

1. Time
2. Category
3. Severity

The following example specifies all three print- options:

```
28-Feb-2000 15:05:32.863 general: notice: running
```

Four predefined channels are used for the BIND server's default logging, as shown in the following example. Section C.5.3.5.2 describes how these channels are used.

```
channel "default_syslog" {
    syslog daemon;                  // send to TCPIP$BIND_RUN.LOG

    severity info;                  // only send priority info
                                    // and higher
};

channel "default_debug" {
    file "TCPIP$BIND_RUN.LOG";      // write to the TCPIP$BIND_RUN.LOG

    severity dynamic;               // log at the server's
                                    // current debug level
};

channel "default_stderr" {
    stderr;                         // write to TCPIP$BIND_RUN.LOG
    severity info;                  // only send priority info
                                    // and higher
};

channel "null" {
   null;                            // discard anything sent to
                                    // this channel
};
```

The default_debug channel only produces output when the server's debug level is not zero. By default, the BIND server writes to the TCPIP$BIND_RUN.LOG file.

Once a channel is defined, it cannot be redefined. Thus, you cannot alter the built-in channels directly, but you can modify the default logging by pointing categories at channels you have defined.

C.5.3.5.2 The Category Phrase There are many categories, so you can send the logs you want to see anywhere, without seeing logs you do not want. If you do not specify a list of channels for a category, then log messages in that category are sent to the default category instead. If you do not specify a default category, the following definition is used:

```
category default { default_syslog; default_debug; };
```

For example, if you want to log security events to a file but you also want to preserve the default logging behavior, specify the following:

```
channel my_security_channel {
    file "my_security_file";
    severity info;
};
category security {
    "my_security_channel";
    default_syslog;
    default_debug;
};
```

To discard all messages in a category, specify the `null` channel:

```
category lame-servers { null; };
category cname { null; };
```

Table C–5 describes the logging categories.

Table C–5 Logging Categories

Category	Meaning
`default`	The logging options for those categories where no specific configuration has been defined.
`general`	The default category for messages that are not classified.
`database`	Messages relating to the databases used internally by the name server to store zone and cache data.
`security`	Approval and denial of requests.
`config`	Configuration file parsing and processing.
`resolver`	DNS resolution, such as the recursive lookups performed on behalf of clients by a caching name server.
`xfer-in`	Zone transfers the server is receiving.
`xfer-out`	Zone transfers the server is sending.
`notify`	The NOTIFY protocol.
`client`	Processing of client requests.
`unmatched`	Messages for which the BIND server was unable to determine the class, or for which there was no matching view. A one-line summary is also logged to the client category.
`network`	Network operations.
`update`	Dynamic updates.
`queries`	Queries. Using this category enables query logging.

(continued on next page)

Table C–5 (Cont.) Logging Categories

Category	Meaning
dispatch	Dispatching of incoming packets to the server modules where they are to be processed.
dnssec	DNSSEC and TSIG protocol processing.
lame-servers	Lame servers (misconfigurations in remote servers, discovered by BIND 9 when trying to query those servers during resolution).

C.5.3.6 The OPTIONS Statement

The options statement sets up global options to be used by BIND. This statement should appear only once in a configuration file. If more than one occurrence is found, the first occurrence determines the actual options used, and a warning is generated. If there is no options statement, an options block with each option set to its default is used.

The options statement has the following syntax:

```
options {
    [ version version_string; ]
    [ directory path_name; ]
    [ named-xfer path_name; ]
    [ tkey-domain domainname; ]
    [ tkey-dhkey key_name key_tag; ]
    [ dump-file path_name; ]
    [ memstatistics-file path_name; ]
    [ pid-file path_name; ]
    [ statistics-file path_name; ]
    [ zone-statistics yes_or_no; ]
    [ auth-nxdomain yes_or_no; ]
    [ deallocate-on-exit yes_or_no; ]
    [ dialup dialup_option; ]
    [ fake-iquery yes_or_no; ]
    [ fetch-glue yes_or_no; ]
    [ has-old-clients yes_or_no; ]
    [ host-statistics yes_or_no; ]
    [ minimal-responses yes_or_no; ]
    [ multiple-cnames yes_or_no; ]
    [ notify yes_or_no | explicit; ]
    [ recursion yes_or_no; ]
    [ rfc2308-type1 yes_or_no; ]
    [ use-id-pool yes_or_no; ]
    [ maintain-ixfr-base yes_or_no; ]
    [ forward ( only | first ); ]
    [ forwarders { ip_addr [port ip_port] ; [ ip_addr [port ip_port] ;
... ] }; ]
    [ check-names ( master | slave |  response )( warn | fail | ignore
); ]
    [ allow-notify { address_match_list }; ]
    [ allow-query { address_match_list }; ]
    [ allow-transfer { address_match_list }; ]
    [ allow-recursion { address_match_list }; ]
    [ allow-v6-synthesis { address_match_list }; ]
    [ blackhole { address_match_list }; ]
    [ listen-on [ port ip_port ] { address_match_list }; ]
    [ listen-on-v6 [ port ip_port ] { address_match_list }; ]
    [ query-source [ address ( ip_addr | * ) ] [ port ( ip_port | * ) ];
]
    [ max-transfer-time-in number; ]
    [ max-transfer-time-out number; ]
    [ max-transfer-idle-in number; ]
```

```
    [ max-transfer-idle-out number; ]
    [ tcp-clients number; ]
    [ recursive-clients number; ]
    [ serial-query-rate number; ]
    [ serial-queries number; ]
    [ transfer-format ( one-answer | many-answers ); ]
    [ transfers-in  number; ]
    [ transfers-out number; ]
    [ transfers-per-ns number; ]
    [ transfer-source (ip4_addr | *) [port ip_port] ; ]
    [ transfer-source-v6 (ip6_addr | *) [port ip_port] ; ]
    [ notify-source (ip4_addr | *) [port ip_port] ; ]
    [ notify-source-v6 (ip6_addr | *) [port ip_port] ; ]
    [ also-notify { ip_addr [port ip_port] ; [ ip_addr [port ip_port] ;
... ] }; ]
    [ max-ixfr-log-size number; ]
    [ cleaning-interval number; ]
    [ heartbeat-interval number; ]
    [ interface-interval number; ]
    [ statistics-interval number; ]
    [ topology { address_match_list }];
    [ sortlist { address_match_list }];
    [ rrset-order { order_spec ; [ order_spec ; ... ] ] };
    [ lame-ttl number; ]
    [ max-ncache-ttl number; ]
    [ max-cache-ttl number; ]
    [ sig-validity-interval number ; ]
    [ min-roots number; ]
    [ use-ixfr yes_or_no ; ]
    [ treat-cr-as-space yes_or_no ; ]
    [ min-refresh-time number ; ]
    [ max-refresh-time number ; ]
    [ min-retry-time number ; ]
    [ max-retry-time number ; ]
    [ port ip_port; ]
    [ additional-from-auth yes_or_no ; ]
    [ additional-from-cache yes_or_no ; ]
    [ random-device path_name ; ]
    [ max-cache-size size_spec ; ]
    [ match-mapped-addresses yes_or_no; ]
};
```

Table C–6 through Table C–14 describe the BIND server configuration options.

Table C–6 BIND Server Configuration Options

Option	Description
version	The version the server should report using a query of name version.bind in class CHAOS. The default is the real version number of this server.
directory	The working directory of the server. Any nonabsolute path names in the configuration file is assumed to be relative to this directory. The default location for the server output file (TCPIP$BIND_RUN.LOG) is this directory. If a directory is not specified, the working directory defaults to SYS$SPECIFIC:[TCPIP$BIND]. If you are configuring a BIND failover environment in an OpenVMS Cluster, the working directory is defined by the logical TCPIP$BIND_COMMON.

(continued on next page)

Table C–6 (Cont.) BIND Server Configuration Options

Option	Description
named-xfer	This option is obsolete. It was used in BIND 8 to specify the path name to the named-xfer program. In BIND 9, no separate named-xfer program is needed; its functionality is built into the name server.
tkey-domain	The domain appended to the names of all shared keys generated with TKEY. When a client requests a TKEY exchange, it may or may not specify the desired name for the key. If present, the name of the shared key is formatted as follows: "*client specified part*" + "*tkey-domain*" If it is not present, the name of the shared key is formatted as follows: "*random hex digits*" + "*tkey-domain*" In most cases, the domain name should be the server's domain name.
tkey-dhkey	The Diffie-Hellman key used by the server to generate shared keys with clients using the Diffie-Hellman mode of TKEY. The server must be able to load the public and private keys from files in the working directory. In most cases, the key name should be the server's host name.
dump-file	The path name of the file the server dumps the database to when instructed to do so with the rndc dumpdb command. If not specified, the default is TCPIP$BIND_DUMP.DB.
memstatistics-file	The path name of the file the server writes memory usage statistics to on exit. If not specified, the default is TCPIP$BIND.MEMSTATS. This option is not yet implemented.
pid-file	The path name of the file in which the server writes its process ID. If the path name is not specified, the default is TCPIP$BIND.PID. The PID file is used by programs that want to send signals to the running name server.
statistics-file	The path name of the file to which the server appends statistics when instructed to do so with the rndc stats command. If not specified, the default is TCPIP$BIND.STATS in the server's current directory. The format of the file is described in Section C.5.3.6.14.
port	The UDP/TCP port number the server uses for receiving and sending DNS protocol traffic. The default is 53. This option is intended mainly for server testing; a server using a port other than 53 cannot communicate with the global DNS. The port option should be placed at the beginning of the options block, before any other options that take port numbers or IP addresses, to ensure that the port value takes effect for all addresses used by the server.

(continued on next page)

Table C–6 (Cont.) BIND Server Configuration Options

Option	Description
random-device	The source of entropy to be used by the server. Entropy is needed primarily for DNSSEC operations, such as TKEY transactions and dynamic update of signed zones. This option specifies the file from which to read entropy. Operations requiring entropy fail when the file has been exhausted. If this option is not specified, entropy does not take place. The random-device option takes effect during the initial configuration load at server startup time and is ignored on subsequent reloads.

C.5.3.6.1 Boolean Options Table C–7 describes the Boolean BIND server configuration options.

Table C–7 BIND Server Boolean Configuration Options

Option	Description
auth-nxdomain	If YES, then the AA bit is always set on NXDOMAIN responses, even if the server is not actually authoritative. The default is NO. This is a change from BIND Version 8. If you are upgrading from old software, you might need to set this option to YES.
deallocate-on-exit	This option was used in BIND Version 8 to enable checking for memory leaks on exit. BIND Version 9 ignores this option and always performs the checks.

(continued on next page)

Table C–7 (Cont.) BIND Server Boolean Configuration Options

Option	Description
dialup	If YES, then the server treats all zones as if they are doing zone transfers across a dial-on-demand dialup link, which can be brought up by traffic originating from this server. This has different effects according to zone type, and it concentrates the zone maintenance so that it all happens in a short interval, once every heartbeat-interval and during the one call. It also suppresses some of the normal zone maintenance traffic. The default is NO. The dialup option can also be specified in the view and zone statements. In these cases, it overrides the global dialup option. If the zone is a master zone, the server sends out a NOTIFY request to all the slaves. This triggers the zone serial number check in the slave (providing it supports NOTIFY), allowing the slave to verify the zone while the connection is active. If the zone is a slave or stub zone, then the server suppresses the regular "zone up to date" (refresh) queries and performs them only when the heartbeat-interval expires, in addition to sending NOTIFY requests. Finer control can be achieved by using the following options: • notify, which sends only NOTIFY messages. • notify-passive, which sends NOTIFY messages and suppresses the normal refresh queries. • refresh, which suppresses normal refresh processing and sends refresh queries when the heartbeat-interval expires. • passive, which disables normal refresh processing.
fake-iquery	In BIND Version 8, this option was used to enable simulating the obsolete DNS query type IQUERY. BIND Version 9 never does IQUERY simulation.
fetch-glue	This option is obsolete. In BIND Version 8, this option caused the server to attempt to fetch glue resource records it lacked when constructing the additional data section of a response. In BIND Version 9, the server does not fetch glue resource records.
has-old-clients	This option was incorrectly implemented in BIND Version 8 and is ignored by BIND Version 9.
host-statistics	In BIND Version 8, this option enabled the keeping of statistics for every host with which the name server interacts. This option is not implemented in BIND Version 9.
maintain-ixfr-base	This option is obsolete. It was used in BIND Version 8 to determine whether a transaction log was kept for incremental zone transfers. BIND Version 9 maintains a transaction log whenever possible. To disable outgoing incremental zone transfers, set the provide-ixfr option to NO. See Section C.5.3.7 for more information.

(continued on next page)

Table C–7 (Cont.) BIND Server Boolean Configuration Options

Option	Description
minimal-responses	Specifies that when the server generates responses, it adds records to the authority and additional data sections only when they are required (for example, for delegations and negative responses). This might improve the performance of the server. The default is NO.
multiple-cnames	This option was used in BIND Version 8 to allow a domain name to allow multiple CNAME records in violation of the DNS standards. BIND Version 9 strictly enforces the CNAME rules, both in master files and dynamic updates.
notify	Sends DNS NOTIFY messages when a zone changes for which the server is authoritative (see Section C.5.5). The messages are sent to the servers listed in the zone's NS records (except the master server identified in the SOA MNAME field) and to any servers listed in the also-notify option. If this option is explicitly set (the default), notifications are sent only to servers explicitly listed using also-notify. If it is set to NO, notifications are not sent. The notify option can also be specified in the zone statement. This overrides the notify option in the options statement.
recursion	When a DNS query requests recursion, specifies that the server will attempt to do all the work required to answer the query. If the recursion option is off and the server does not already know the answer, it returns a referral response. The default is YES. Note that setting the recursion option to NO does not prevent clients from getting data from the server's cache; it only prevents new data from being cached as an effect of client queries. Caching can still occur as an effect of the server's internal operation, such as NOTIFY address lookups.
rfc2308-type1	Setting this option to YES causes the server to send NS records along with the SOA record for negative answers. The default is NO. This option is not yet implemented.
use-id-pool	This option is obsolete. BIND Version 9 always allocates query IDs from a pool.
zone-statistics	Collects statistical data on all zones in the server. These statistics can be accessed using the rndc stats command, which dumps them to the file listed in the statistics-file option. See Section C.10 for more information.
use-ixfr	This option is obsolete. If you need to disable IXFR to a particular server, see the information about the provide-ixfr option in Section C.5.3.7.
treat-cr-as-space	This option was used in BIND 8 to make the server treat carriage return characters the same way as a space or tab character—to facilitate loading of zone files. In BIND 9, these characters are always accepted and the option is ignored.

(continued on next page)

Table C–7 (Cont.) BIND Server Boolean Configuration Options

Option	Description
additional-from-auth additional-from-cache	These options control the behavior of an authoritative server when answering queries that have additional data or when following CNAME and DNAME chains. When both of these options are set to YES (the default) and a query is being answered from authoritative data (a zone configured into the server), the additional data section of the reply is filled in using data from other authoritative zones and from the cache. In some situations this is undesirable, such as when there is concern over the correctness of the cache, or in servers where slave zones can be added and modified by untrusted third parties. Also, avoiding the search for this additional data speeds up server operations at the possible expense of additional queries to resolve what otherwise would be provided in the additional section. For example, if a query asks for an MX record for host FOO.EXAMPLE.COM, the following record is found: `MX 10 mail.example.net` The address records (A, A6, and AAAA) for MAIL.EXAMPLE.NET are provided as well, if they are known. Setting these options to NO disables this behavior. These options are intended for use in authoritative-only servers or in authoritative-only views. If you attempt to set these options to NO without also specifying `recursion no`, the server ignores the options and log a warning message. Specifying `additional-from-cache no` disables the use of the cache not only for additional data lookups, but also when looking up the answer. This is usually the desired behavior in an authoritative-only server where the correctness of the cached data is an issue. When a name server is nonrecursively queried for a name that is not below the apex of any served zone, it normally answers with an "upward referral" to the root servers or to the servers of some other known parent of the query name. Because the data in an upward referral comes from the cache, the server cannot provide upward referrals when `additional-from-cache no` has been specified. Instead, the server responds to such queries with "REFUSED." This should not cause any problems, because upward referrals are not required for the resolution process.
match-mapped-addresses	When this option is set, an IPv4-mapped IPv6 address matches any address match list entries that match the corresponding IPv4 address. Use of this option is not necessary on OpenVMS systems.

C.5.3.6.2 Forwarding Options The forwarding facility helps you create a large, sitewide cache on a few servers, thereby reducing traffic over links to external name servers. It can also be used to allow queries by servers that do not have direct access to the Internet but that want to look up exterior names anyway. Forwarding occurs only on those queries for which the server is not authoritative and does not have the answer in its cache.

Table C–8 describes the forwarding options.

Table C–8 Forwarding Options

Option	Description
forward	Meaningful only if the forwarders list is not empty. A value of first (the default) causes the server to query the forwarders first, and if that does not answer the question, the server then looks for the answer itself. If only is specified, the server queries only the forwarders.
forwarders	Specifies the IP addresses to be used for forwarding. The default is the empty list (no forwarding).

Forwarding can also be configured on a per-domain basis, allowing for the global forwarding options to be overridden in a variety of ways. You can set particular domains to use different forwarders, or have a different forward only /first behavior, or not to forward at all. See Section C.5.3.10 for more information.

C.5.3.6.3 Access Control Options Access to the server can be restricted based on the IP address of the requesting system. See Section C.5.2 for details on how to specify IP address lists.

Table C–9 describes the access control options.

Table C–9 Access Control Options

Option	Description
allow-notify	Specifies which hosts are allowed to notify slaves of a zone change in addition to the zone masters. The allow-notify option can also be specified in the zone statement; in this case, it overrides the allow-notify option in the options statement. The allow-notify option is meaningful only for a slave zone. If this option is not specified, the default is to process notify messages from only a zone's master.
allow-query	Specifies which hosts are allowed to ask ordinary questions. The allow-query option can also be specified in the zone statement; in this case, it overrides the allow-query option in the options statement. If this option is not specified, the default is to allow queries from all hosts.
allow-recursion	Specifies which hosts are allowed to make recursive queries through this server. If this option is not specified, the default is to allow recursive queries from all hosts. Note that disallowing recursive queries for a host does not prevent the host from retrieving data that is already in the server's cache.
allow-v6-synthesis	Specifies which hosts are to receive synthetic responses to IPv6 queries, as described in Section C.5.3.6.12.

(continued on next page)

Table C–9 (Cont.) Access Control Options

Option	Description
allow-transfer	Specifies which hosts are allowed to receive zone transfers from the server. The allow-transfer option can also be specified in the zone statement; in this case, it overrides the allow-transfer statement in the options statment. If this option is not specified, the default is to allow transfers from all hosts.
blackhole	Specifies a list of addresses from which the server will not accept queries or will not use to resolve a query. The server will not respond queries from these addresses. The default is NONE.

C.5.3.6.4 Interfaces Options The interfaces and ports from which the server answers queries can be specified using the listen-on options. Table C–10 describes the listen-on options.

Table C–10 Interfaces Options

Option	Description

listen-on	Specifies the port for listening for queries sent using IPv4 addresses.
	The listen-on option takes an optional port number and an address_match_list. The server listens on all interfaces allowed by the address match list. If a port is not specified, port 53 is used.
	Multiple listen-on statements are allowed. For example:
	`listen-on { 5.6.7.8; };`
	`listen-on port 1234 { !1.2.3.4; 1.2/16; };`
	These statements enable the name server on port 53 for the IP address 5.6.7.8, and on port 1234 of an address on the machine in net 1.2 that is not 1.2.3.4.
	If the listen-on option is not specified, the server listens on port 53 on all interfaces.

(continued on next page)

Table C–10 (Cont.) Interfaces Options

Option	Description

listen-on-v6	Specifies the ports on which the server listens for incoming queries sent using IPv6. The server does not bind a separate socket to each IPv6 interface address as it does for IPv4. Instead, it always listens on the IPv6 wildcard address. Therefore, the values allowed for the address_match_list argument to the listen-on-v6 option are:
	• any
	• none
	Multiple listen-on-v6 options can be used to listen on multiple ports. For example:
	`listen-on-v6 port 53 { any; };`
	`listen-on-v6 port 1234 { any; };`
	To make the server not listen on any IPv6 address, specify the following:
	`listen-on-v6 { none; };`
	If the listen-on-v6 option is not specified, the server does not listen on any IPv6 address.

C.5.3.6.5 The Query Address Options If the server does not know the answer to a question, it queries other name servers. The query address options allow you to specify the address and port for these queries.

Table C–11 describes the query address options.

Table C–11 Query Address Options

Option	Description
query-source	Specifies the IPv4 address and port used for such queries. If the address is a wildcard character or is omitted, a wildcard IP address (INADDR_ANY) is used. If the port is a wildcard character or is omitted, a random unprivileged port is used. The default is: `query-source address * port *;`
query-source-v6	Specifies the IPv6 address and port used for such queries. The default is: `query-source-v6 address * port *`

The address specified in the query-source option is used for both UDP and TCP queries, but the port applies only to UDP queries. TCP queries always use a random, unprivileged port.

C.5.3.6.6 Zone Transfer Options BIND includes mechanisms to facilitate zone transfers and to limit the amount of load that transfers place on the system. Table C–12 describes the zone transfer options.

Table C–12 Zone Transfer Options

Option	Description
`also-notify`	Defines a global list of IP addresses of name servers that are also sent NOTIFY messages whenever a fresh copy of the zone is loaded, in addition to the servers listed in the zone's NS records. This helps to ensure that copies of the zones will quickly converge on stealth servers. If an `also-notify` list is given in a `zone` statement, that list overrides the `also-notify` options in the `options` statement. When a zone `notify` statement is set to NO, the IP addresses in the global `also-notify` list are not sent NOTIFY messages for that zone. The default is the empty list (no global notification list).
`max-transfer-time-in`	Inbound zone transfers running longer than this many minutes are terminated. The default is 120 minutes.
`max-transfer-idle-in`	Inbound zone transfers making no progress in this many minutes are terminated. The default is 60 minutes.
`max-transfer-time-out`	Outbound zone transfers running longer than this many minutes are terminated. The default is 120 minutes.
`max-transfer-idle-out`	Outbound zone transfers making no progress in this many minutes are terminated. The default is 60 minutes.
`serial-query-rate`	Slave servers periodically query master servers to find out whether zone serial numbers have changed. Each such query uses a minute amount of the slave server's network bandwidth. To limit the amount of bandwidth used, BIND 9 limits the rate at which queries are sent. The value of the `serial-query-rate` option is the maximum number of queries sent per second. The default is 20.
`serial-queries`	In BIND 8, this option set the maximum number of concurrent serial number queries allowed to be outstanding at any given time. BIND 9 does not limit the number of outstanding serial queries and ignores the `serial-queries` option. Instead, it limits the rate at which the queries are sent as defined by the `serial-query-rate` option.
`transfer-format`	Specifies whether zone transfers are sent using the one-answer format or the many-answers format. The `transfer-format` option is used on the master server to determine which format it sends. When set to `one-answer`, it uses one DNS message per resource record transferred. When set to `many-answers`, it packs as many resource records as possible into a message. `many-answers` is more efficient, but it is supported only by relatively new slave servers, such as BIND Version 9, BIND Version 8, and later versions of BIND Version 4. The default is `many-answers`. The `transfer-format` option can be overridden on a per-server basis by using the `server` statement.

(continued on next page)

Table C–12 (Cont.) Zone Transfer Options

Option	Description
transfers-in	Specifies the maximum number of inbound zone transfers that can be running concurrently. The default value is 10. Increasing the transfers-in value might speed up the convergence of slave zones, but it also might increase the load on the local system.
transfers-out	Specifies the maximum number of outbound zone transfers that can be running concurrently. Zone transfer requests in excess of the limit are refused. The default value is 10.
transfers-per-ns	Specifies the maximum number of inbound zone transfers that can be concurrently transferring from a given remote name server. The default value is 2. Increasing the value of the transfers-per-ns option might speed up the convergence of slave zones, but it also might increase the load on the remote name server. This option can be overridden on a per-server basis by using the transfers phrase of the server statement.
transfer-source	Determines which local address is bound to IPv4 TCP connections used to fetch zones transferred inbound by the server. It also determines the source IPv4 address and, optionally, the UDP port used for the refresh queries and forwarded dynamic updates. If not set, this option defaults to a system-controlled value, which is usually the address of the interface closest to the remote end. This address must appear in the remote end's allow-transfer option for the zone being transferred, if one is specified. This statement sets the transfer source for all zones, but it can be overridden on a per-view or per-zone basis by including a transfer-source statement within the view or zone statement in the configuration file.
transfer-source-v6	Determines which local address is bound to IPv6 TCP connections used to fetch zones transferred inbound by the server. This is the same as the transfer-source option, except zone transfers are performed using IPv6.
notify-source	Determines which local source address and, optionally, UDP port is used to send NOTIFY messages. This address must appear in the slave server's masters clause in the zone statement or in an allow-notify clause. This statement sets the notify-source for all zones, but it can be overridden on a per-zone or per-view basis by including a notify-source statement within the zone or view statement in the configuration file.
notify-source-v6	Determines which local source address and, optionally, UDP port is used to send NOTIFY messages. This option is identical to notify-source, but it applies to NOTIFY messages sent to IPv6 addresses.

C.5.3.6.7 Server Resource Limits Table C–13 describes options that limit the server's resource consumption and are enforced internally by the server rather than by the operating system.

Table C–13 Server Resource Limit Options

Option	Description
max-ixfr-log-size	This option is obsolete; it is accepted and ignored.
recursive-clients	Specifies the maximum number of simultaneous recursive lookups the server performs on behalf of clients. The default is 1000. Because each recursing client uses about 20 kilobytes of memory, the value of the recursive-clients option might have to be decreased on hosts with limited memory.
tcp-clients	Specifies the maximum number of simultaneous client TCP connections that the server accepts. The default is 100.
max-cache-size	Specifies the maximum amount of memory (in bytes) to use for the server's cache. When the amount of data in the cache reaches this limit, the server causes records to expire prematurely so that the limit is not exceeded. In a server with multiple views, the limit applies separately to the cache of each view. The default is unlimited, which means that records are purged from the cache only when their TTL (time-to-live) values expire.

C.5.3.6.8 Periodic Task Intervals Options Table C–14 describes the options that control the intervals for periodic tasks.

Table C–14 Periodic Task Intervals Options

Option	Description
cleaning-interval	The server removes expired resource records from the cache every cleaning-interval minutes. The default is 60 minutes. If set to 0, periodic cleaning does not occur.
heartbeat-interval	The server performs zone maintenance tasks for all dialup zones whenever this interval expires. The default is 60 minutes. The maximum value is one day (1440 minutes). If this option is set to 0, zone maintenance for these zones does not occur.
interface-interval	The server scans the network interface list every interface-interval minutes. The default is 60 minutes. If this option is set to 0, interface scanning occurs only when the configuration file is loaded. After the scan, listeners are started on any new interfaces (provided they are allowed by the listen-on configuration). Listeners on interfaces that have gone away are cleaned up.
statistics-interval	Name server statistics are logged every statistics-interval minutes. The default is 60. If set to 0, statistics are not logged. This option is not yet implemented.

C.5.3.6.9 The TOPOLOGY Statement When the server chooses a name server to query from a list of name servers, it prefers the one that is topologically closest to itself. The topology statement takes an address match list and interprets it in a special way. Each top-level list element is assigned a distance. Nonnegated elements get a distance based on their position in the list; the closer the match is to the start of the list, the shorter the distance between it and the server. A negated match is assigned the maximum distance from the server. If there is

no match, the address gets a distance that is further than any nonnegated list element and closer than any negated element. For example:

```
topology {
    10/8;
    !1.2.3/24;
    { 1.2/16; 3/8; };
};
```

The example configuration prefers servers on network 10 the most, followed by hosts on network 1.2.0.0 (netmask 255.255.0.0) and network 3, with the exception of hosts on network 1.2.3 (netmask 255.255.255.0), which is the least preferred. The default topology is:

```
topology { localhost; localnets; };
```

Note

The `topology` statement is not implemented in BIND Version 9.

C.5.3.6.10 The SORTLIST Statement The response to a DNS query can consist of multiple resource records (RRs) forming a resource record set (RRset). The name server normally returns the RRs within the RRset in an indeterminate order. (See Section C.5.3.6.11.)

The client resolver code should rearrange the RRs as appropriate, that is, by using any addresses on the local network in preference to other addresses. However, not all resolvers can do this or are correctly configured. When a client is using a local server the sorting can be performed in the server, based on the client's address. This requires configuring only the name servers, not all the clients.

The `sortlist` statement takes an address match list and interprets it even more specifically than the `topology` statement does (see Section C.5.3.6.9). Each top-level statement in the `sortlist` must itself be an explicit address match list with one or two elements. The first element (which can be an IP address, an IP prefix, an ACL name, or a nested address match list) of each top-level list is checked against the source address of the query until a match is found.

Once the source address of the query is matched, if the top-level statement contains only one element, the actual primitive element that matched the source address is used to select the address in the response to move to the beginning of the response. If the statement is a list of two elements, then the second element is treated the same as the address match list in a `topology` statement. Each top-level element is assigned a distance and the address in the response with the minimum distance is moved to the beginning of the response.

Example 1

In the following example, any queries received from any of the addresses of the host itself gets responses that prefer addresses on any of the locally connected networks. The next-most-preferred are addresses on the 192.168.1/24 network, and after that either the 192.168.2/24 or 192.168.3/24 network with no preference shown between these two networks. Queries received from a host on the 192.168.1/24 network prefers other addresses on that network to the 192.168.2/24 and 192.168.3/24 networks. Queries received from a host on the 192.168.4/24 or the 192.168.5/24 network prefer only other addresses on their directly connected networks.

```
sortlist {
    { localhost;                        // IF the local host
        { localnets;                    // THEN first fit on the
            192.168.1/24;               // following nets
            { 192.168.2/24; 192.168.3/24; }; }; };
    { 192.168.1/24;                     // IF on class C 192.168.1
        { 192.168.1/24;                 // THEN use .1, or .2 or .3
            { 192.168.2/24; 192.168.3/24; }; }; };
    { 192.168.2/24;                     // IF   on class C 192.168.2
        { 192.168.2/24;                 // THEN use .2, or .1 or .3
            { 192.168.1/24; 192.168.3/24; }; }; };
    { 192.168.3/24;                     // IF on class C 192.168.3
        { 192.168.3/24;                 // THEN use .3, or .1 or .2
            { 192.168.1/24; 192.168.2/24; }; }; };
    { { 192.168.4/24; 192.168.5/24; };  // if .4 or .5, prefer that net
    };
};
```

Example 2

The following example illustrates reasonable behavior for the local host and for hosts on directly connected networks. This behavior is similar to that of the address sort in BIND Version 4. Responses sent to queries from the local host favor any of the directly connected networks. Responses sent to queries from any other hosts on a directly connected network prefer addresses on that same network. Responses to other queries are not sorted.

```
sortlist {
          { localhost; localnets; };
          { localnets; };
};
```

C.5.3.6.11 RRset Ordering When multiple records are returned in an answer, it might be useful to configure the order of the records placed into the response. The `rrset-order` statement permits configuration of the ordering of the records in a multiple-record response.

An `order_spec` is defined as follows:

```
[ class class_name ][ type type_name ][ name "domain_name"]
 order ordering
```

If no class is specified, the default is ANY. If no type is specified, the default is ANY. If no name is specified, the default is wildcard.

The legal values for *ordering* are:

- `fixed` (Records are returned in the order they are defined in the zone file.)
- `random` (Records are returned in random order.)
- `cyclic` (Records are returned in round-robin order.)

For example:

```
rrset-order {
   class IN type A name "host.example.com" order random;
   order cyclic;
};
```

This example causes any responses for type A records in class IN that have `host.example.com` as a suffix to always be returned in random order. All other records are returned in cyclic order.

If multiple rrset-order statements appear, they are not combined; the last one applies.

Note

The rrset-order statement is not yet implemented. BIND currently supports only "random-cyclic" ordering, in which the server randomly chooses a starting point within the RRset and returns the records in order starting at that point, wrapping around the end of the RRset if necessary.

C.5.3.6.12 Synthetic IPv6 Responses Many existing stub resolvers support IPv6 DNS lookups as defined in RFC 1886, using AAAA records for forward lookups and nibble labels in the ip6.int domain for reverse lookups. They do not support RFC 2874 lookups (using A6 records and binary labels in the ip6.arpa domain).

To continue to use such stub resolvers, BIND Version 9 provides a way to automatically convert RFC 1886 lookups into RFC 2874 lookups and to return the results as "synthetic" AAAA and PTR records.

This feature is disabled by default and can be enabled on a per-client basis by adding the following clause to the to the options or view statement:

```
allow-v6-synthesis { address_match_list };
```

When this feature is enabled, recursive AAAA queries cause the server first to try an A6 lookup and then, if that fails, an AAAA lookup. Regardless of which one succeeds, the results are returned as a set of synthetic AAAA records. Similarly, recursive PTR queries in ip6.int cause a lookup in ip6.arpa using binary labels and, if that fails, another lookup in ip6.int. The results are returned as a synthetic PTR record in ip6.int.

The synthetic records have a TTL value of 0. DNSSEC validation of synthetic responses is not supported; therefore, responses containing synthetic RRs do not have the AD flag set.

C.5.3.6.13 Tuning Options Table C–15 describes the options provided for tuning the BIND server.

Table C–15 Tuning Options

Options	Description
lame-ttl	Sets the number of seconds to cache a lame server indication. A value of zero disables caching. (This is not recommended.) The default is 600 (10 minutes); the maximum value is 1800 (30 minutes).
max-ncache-ttl	To reduce network traffic and increase performance, the server stores negative answers. The max-ncache-ttl option is used to set a maximum retention time for these answers in the server in seconds. The default is 10800 seconds (3 hours). The value of max-ncache-ttl cannot exceed 7 days and is silently truncated to 7 days if set to a greater value.

(continued on next page)

Table C–15 (Cont.) Tuning Options

Options	Description
max-cache-ttl	Sets the maximum time for which the server caches ordinary (positive) answers. The default is one week (7 days).
min-roots	The minimum number of root servers that is required for a request for the root servers to be accepted. The default is 2. This option is not yet implemented.
sig-validity-interval	Specifies the number of days into the future when DNSSEC signatures automatically generated as a result of dynamic updates will expire. (See Section C.5.7 for more information.) The default is 30 days. The signature inception time is unconditionally set to one hour before the current time to allow for a limited amount of clock skew.
min-refresh-time max-refresh-time min-retry-time max-retry-time	Controls the server's behavior when refreshing a zone (querying for SOA changes) or when retrying failed transfers. Usually the SOA values for the zone are used, but these values are set by the master, giving slave server administrators little control over their contents. These options allow the administrator to set a minimum and maximum refresh and retry time on a per-zone, per-view, or per-server basis. These options are valid for master, slave, and stub zones, and they set the SOA refresh and retry times to the specified values.

C.5.3.6.14 The Statistics File The statistics file generated by BIND 9 is similar, but not identical, to that generated by BIND 8.

The statistics dump begins with the following line:

```
+++ Statistics Dump +++ (973798949)
```

The number in parentheses is a standard UNIX time stamp, measured as seconds since January 1, 1970. Following that line are a series of lines containing a counter type, the value of the counter, a zone name (optional), and a view name (optional). The lines without view and zone listed are global statistics for the entire server. Lines with a zone and view name apply to the given view and zone (the view name is omitted for the default view). The statistics dump ends with the following line:

```
--- Statistics Dump --- (973798949)
```

The time stamp is identical to the one in the beginning line.

Table C–16 describes the statistics counters that are maintained.

Table C–16 Statistics Counters

Counter	Description
success	The number of successful queries made to the server or zone. A successful query is defined as query that returns a NOERROR response other than a referral response.
referral	The number of queries that resulted in referral responses.

(continued on next page)

Table C–16 (Cont.) Statistics Counters

Counter	Description
nxrrset	The number of queries that resulted in NOERROR responses with no data.
nxdomain	The number of queries that resulted in NXDOMAIN responses.
recursion	The number of queries that caused the server to perform recursion in order to find the final answer.
failure	The number of queries that resulted in a failure response other than those described in the preceding counters.

C.5.3.7 The SERVER Statement

The server statement defines characteristics to be associated with a remote name server. The server statement has the following syntax:

```
server ip_addr {
    [ bogus yes_or_no ; ]
    [ provide-ixfr yes_or_no ; ]
    [ request-ixfr yes_or_no ; ]
    [ edns yes_or_no ; ]
    [ transfers number ; ]
    [ transfer-format ( one-answer | many-answers ) ; ]]
    [ keys { string ; [ string ; [...]] } ; ]
};
```

The server statement can occur at the top level of the configuration file or inside a view statement. If a view statement contains one or more server statements, only those apply to the view, and any top-level ones are ignored. If a view contains no server statements, any top-level server statements are used as defaults.

Table C–17 describes the clauses in the server statement.

Table C–17 Server Statement Clauses

Clause	Description
bogus	If you discover that a remote server is giving out bad data, marking it as bogus prevents further queries to it. The default value of bogus is NO.
provide-ixfr	Determines whether the local server, acting as master, responds with an incremental zone transfer when the given remote server, a slave, requests it. If this option is set to YES, incremental transfer is provided whenever possible. If set to NO, all transfers to the remote server are nonincremental. If not set, the value of the provide-ixfr option in the global options statement is used as a default.

(continued on next page)

Table C–17 (Cont.) Server Statement Clauses

Clause	Description
request-ixfr	Determines whether the local server, acting as a slave, requests incremental zone transfers from the given remote server, a master. If this option is not set, the value of the request-ixfr option in the global options statement is used as a default. IXFR requests to servers that do not support IXFR automatically falls back to AXFR. Therefore, you do not need to list manually which servers support IXFR and which ones do not; the global default of YES should always work. The purpose of the provide-ixfr and request-ixfr clauses is to make it possible to disable the use of IXFR, even when both master and slave claim to support it; for example, if one of the servers crashes or corrupts data when IXFR is used. See Section C.5.6 for more information.
edns	Determines whether the local server attempts to use EDNS when communicating with the remote server. The default is YES.
transfer-format	Specifies the zone transfer method: • one-answer uses one DNS message per resource record transferred. • many-answers packs as many resource records as possible into a message. The many-answers mode is more efficient, but it is understood only by BIND Version 9, BIND Version 8, and later versions of BIND Version 4. If transfer-format is not specified, the transfer format specified by the options statement is used.
transfers	Limits the number of concurrent inbound zone transfers from the specified server. If no transfers clause is specified, the limit is set according to the transfers-per-ns option, as described in Table C–12.
keys	Specifies a key_id defined by the key statement, to be used for transaction security when talking to the remote server. The key statement must come before the server statement that references it. When a request is sent to the remote server, a request signature is generated using the key specified here and appended to the message. A request originating from the remote server is not required to be signed by this key. Use only one key for each server.

C.5.3.8 The TRUSTED-KEYS Statement

The trusted-keys statement defines DNSSEC security roots. (DNSSEC is described in Section C.2.6.)

A security root is defined when the public key for a nonauthoritative zone is known but cannot be securely obtained through DNS, either because it is the DNS root zone or because its parent zone is unsigned. Once a key has been configured as a trusted key, it is treated as if it had been validated and proven secure. The resolver attempts DNSSEC validation on all DNS data in subdomains of a security root.

The trusted-keys statement can contain multiple key entries. The trusted-keys statement has the following syntax:

```
trusted-keys {
    string number number number string ;
    [ string number number number string ; [...]]
};
```

Each statement contains the key's domain name, flags, protocol, algorithm, and base-64 representation of the key data.

The base-64 representation of the key data must be enclosed in quotation marks.

C.5.3.9 The VIEW Statement

The view statement allows a name server to answer a DNS query differently, depending on who is asking. It is particularly useful for implementing split DNS setups without having to run multiple servers.

The view statement has the following syntax:

```
view view_name [class] {
      match-clients { address_match_list } ;
      match-destinations { address_match_list } ;
      match-recursive-only { yes_or_no } ;
      [ view_option; ...]
      [ zone-statistics yes_or_no ; ]
      [ zone_statement; ...]
};
```

The parameters to the view statement are:

- *view-name*, which specifies the name of this view.
- *class*, which specifies the class for this view. If no class is given, class IN is assumed.

Note

All non-IN views must contain a hint zone. Only the IN class has compiled-in default hints.

Table C–18 describes the clauses you can include in the view statement.

Table C–18 View Statement Clauses

Clause	Description
match-clients match-destinations	Each view statement defines a view of the DNS name space that is seen by a subset of clients. A client matches a view if its source IP address matches the address match list of the view's match-clients clause and its destination IP address matches the address match list of the view's match-destinations clause. If they are not specified, both match-clients and match-destinations default to matching all addresses.
match-recursive-only	Only recursive requests from matching clients match that view.

(continued on next page)

Table C–18 (Cont.) View Statement Clauses

Clause	Description
view-options	Many of the options given in the options statement can also be used in a view statement, and then apply only when resolving queries with that view. When no view statement value is given, the value in the options statement is used as the default.
zone-statistics	Specifies whether or not to generate the zone statistics file. See Section C.5.3.6.14 for more information.
zone-statement	Specifies the zone information for this view. See Section C.5.3.10 for more information.

The order of the view statements is significant A client request is resolved in the context of the first view that it matches. Zones defined within a view statement are accessible only to clients that match the view.

By defining a zone of the same name in multiple views, different zone data can be given to different clients; for example, internal and external clients in a split DNS setup. Also, zone statement options can have default values specified in the view statement; these view-specific defaults take precedence over those in the options statement.

If there are no view statements in the configuration file, a default view that matches any client is automatically created in class IN, and any zone statements specified on the top level of the configuration file are considered to be part of this default view.

Note

If any explicit view statements are present, all zone statements must occur inside view statements.

The following example shows a typical split DNS setup implemented using view statements:

```
view "internal" {
               // This should match our internal networks.
      match-clients { 10.0.0.0/8; };
               // Provide recursive service to internal clients only.
      recursion yes;
               // Provide a complete view of the example.com zone
               // including addresses of internal hosts.
      zone "example.com" {
            type master;
            file "example-internal.db";
      };
};
view "external" {
      match-clients { any; };
               // Refuse recursive service to external clients.
      recursion no;
               // Provide a restricted view of the example.com zone
               // containing only publicly accessible hosts.
      zone "example.com" {
            type master;
            file "example-external.db";
      };
};
```

C.5.3.10 The ZONE Statement

The zone statement specifies options for a specific zone. Note that if view statements are included in the configuration file, the zone statements must be included in view statements.

The zone statement has the following syntax:

```
zone zone_name [class] [{
    type ( master | slave | hint | stub | forward ) ;
    [ allow-notify { address_match_list } ; ]
    [ allow-query { address_match_list } ; ]
    [ allow-transfer { address_match_list } ; ]
    [ allow-update { address_match_list } ; ]
    [ update-policy { update_policy_rule [...] } ; ]
    [ allow-update-forwarding { address_match_list } ; ]
    [ also-notify { ip_addr [port ip_port] ; [ ip_addr [port ip_port] ;
... ] }; ]
    [ check-names (warn|fail|ignore) ; ]
    [ dialup dialup_option ; ]
    [ file string ; ]
    [ forward (only|first) ; ]
    [ forwarders { ip_addr [port ip_port] ; [ ip_addr [port ip_port] ;
... ] }; ]
    [ ixfr-base string ; ]
    [ ixfr-tmp-file string ; ]
    [ maintain-ixfr-base yes_or_no ; ]
    [ masters [port ip_port] { ip_addr [port ip_port] [key key]; [...] }
; ]
    [ max-ixfr-log-size number ; ]
    [ max-transfer-idle-in number ; ]
    [ max-transfer-idle-out number ; ]
    [ max-transfer-time-in number ; ]
    [ max-transfer-time-out number ; ]
    [ notify yes_or_no | explicit ; ]
    [ pubkey number number number string ; ]
    [ transfer-source (ip4_addr | *) [port ip_port] ; ]
    [ transfer-source-v6 (ip6_addr | *) [port ip_port] ; ]
    [ notify-source (ip4_addr | *) [port ip_port] ; ]
    [ notify-source-v6 (ip6_addr | *) [port ip_port] ; ]
    [ zone-statistics yes_or_no ; ]
    [ sig-validity-interval number ; ]
```

```
    [ database string ; ]
    [ min-refresh-time number ; ]
    [ max-refresh-time number ; ]
    [ min-retry-time number ; ]
    [ max-retry-time number ; ]
};
```

C.5.3.10.1 Type of Zone Table C–19 describes the types of zones that you can specify in the `type` clause.

Table C–19 Zone Types

Type	Description
`master`	The server that has a master copy of the data for the zone and that can provide authoritative answers for it.
`slave`	A replica of a master zone. The `masters` option specifies the IP addresses of one or more master servers that this slave can contact to update its copy of the zone information.
`stub`	Similar to a slave zone, except that it replicates only the NS records of a master zone instead of the entire zone. Stub zones are not a standard part of the DNS; they are a feature specific to the BIND implementation. Stub zones can be used to eliminate the need for glue NS record in a parent zone at the expense of maintaining a stub zone entry and a set of name server addresses in TCPIP$BIND.CONF. This usage is not recommended for new configurations, and BIND Version 9 supports it only in a limited way. In BIND Version 4 and BIND Version 8, zone transfers of a parent zone included the NS records from stub children of that zone. This made it possible to configure child stubs only in the master server for the parent zone. BIND Version 9 never mixes together zone data from different zones in this way. Therefore, if a BIND Version 9 master serving a parent zone has child stub zones, all the slave servers for the parent zone also need to have the same child stub zones. Stub zones can also be used as a way of forcing the resolution of a given domain to use a particular set of authoritative servers. For example, the caching name servers on a private network using RFC 2157 addressing can be configured with stub zones for `10.in-addr.arpa` to use a set of internal name servers as the authoritative servers for that domain.

(continued on next page)

Table C–19 (Cont.) Zone Types

Type	Description
forward	A forward zone allows you to configure forwarding on a per-domain basis. A zone statement of type forward can contain forward and forwarders statements, which applies to queries within the domain specified by the zone name. If no forwarders statement is present or if an empty list for forwarders is specified, then forwarding is not done for the domain, thereby canceling the effects of any forwarders in the options statement. If you want to use this type of zone to change the behavior of the global forward option (using the first value to specify the zone to which to forward first, or the only value to specify forwarding to this zone only), and you want to use the same servers that are set globally, you need to respecify the global forwarders.
hint	The initial set of root name servers is specified using a hint zone. When the server starts up, it uses the root hints to find a root name server and to get the most recent list of root name servers. If no hint zone is specified for class IN, the server uses a default set of root servers hints. Classes other than IN have no built-in defaults hints.

C.5.3.10.2 The Zone Class The zone name can optionally be followed by a class. If the class is not specified, class IN (for Internet) is assumed. This is correct for the vast majority of cases.

The hesiod class is named for an information service from MIT's Project Athena. It is used to share information about various systems databases, such as users, groups, printers, and so on. The keyword HS is a synonym for hesiod.

Another MIT development is CHAOSnet, a LAN protocol created in the mid-1970s. Zone data for CHAOSnet can be specified with the CH class.

C.5.3.10.3 Zone Options Table C–20 describes the options you can include in the zone statement.

Table C–20 Zone Options

Option	Description
allow-notify	See the description of allow-notify in Section C.5.3.6.3.
allow-query	See the description of allow-query in Section C.5.3.6.3.
allow-transfer	See the description of allow-transfer in Section C.5.3.6.3.
allow-update	Specifies which hosts are allowed to submit dynamic DNS updates for master zones. The default is to deny updates from all hosts.
update-policy	Specifies an update policy, as described in Section C.5.7.2.

(continued on next page)

Table C–20 (Cont.) Zone Options

Option	Description
allow-update-forwarding	Specifies which hosts are allowed to submit dynamic updates to slave zones to be forwarded to the master. The default is NONE, which means that update forwarding is not performed. To enable update forwarding, specify ANY. Do not specify any other value; the responsibility for update access control should rest with the master server, not with the slaves. Enabling the update forwarding feature on a slave server can expose master servers relying on insecure IP-address based access control to attacks; see Section C.2.2 for more details.
also-notify	Meaningful only if notify is active for this zone. The set of machines that receives a NOTIFY message for this zone is made up of all the listed name servers (other than the primary master) for the zone, plus any IP addresses specified with the also-notify statement. A port can be specified with each also-notify address to send the notify messages to a port other than the default of 53. also-notify is not meaningful for stub zones. The default is the empty list.
check-names	This option was used in BIND 8 to restrict the character set of domain names in master files and of DNS responses received from the network. BIND 9 does not restrict the character set of domain names and does not implement the check-names option.
database	Specifies the type of database to be used for storing the zone data. The string following the database keyword is interpreted as a list of space-delimited words. The first word identifies the database type; any subsequent words are passed as arguments to the database to be interpreted in a way specific to the database type. The default is rbt, the native database used by BIND 9. This database does not take arguments.
dialup	See the description of the dialup option in Section C.5.3.6.1.
forward	Meaningful only if the zone has a forwarders list. The only keyword causes the lookup to fail after trying the forwarders and getting no answer. The first keyword allows attempts at normal lookups.
forwarders	Overrides the list of global forwarders. If the zone type is not forward, forwarding is not done for the zone, and the global options are not used.
ixfr-base	This option was used in BIND 8 to specify the name of the transaction log (journal) file for dynamic update and IXFR. BIND 9 ignores this option and constructs the name of the journal file by appending _JNL to the name of the zone file.
ixfr-tmp-file	An undocumented option in BIND 8. Ignored in BIND 9.

(continued on next page)

Table C–20 (Cont.) Zone Options

Option	Description
masters	Specifies one or more IP addresses of master servers that the slave contacts to update its copy of the zone. By default, transfers are made from port 53 on the servers; this can be changed for all servers by specifying a port number before the list of IP addresses, or on a per-server basis after the IP address. Authentication to the master can also be done with per-server TSIG keys. If a file is specified, then the replica is written to this file whenever the zone is changed and is reloaded from this file on a server restart. Use of a file is recommended because it often speeds server startup and eliminates a waste of bandwidth.
max-transfer-time-in	See the description of max-transfer-time-in in Section C.5.3.6.6.
max-transfer-idle-in	See the description of max-transfer-idle-in in Section C.5.3.6.6.
max-transfer-time-out	See the description of max-transfer-time-out in Section C.5.3.6.6.
max-transfer-idle-out	See the description of max-transfer-idle-out in Section C.5.3.6.6.
notify	See the description of notify in Section C.5.3.6.
pubkey	In BIND Version 8, this option was used for specifying a public zone key for verification of signatures in DNSSEC-signed zones when they are loaded from disk. BIND Version 9 does not verify signatures on loading and ignores the option.
zone-statistics	If YES, the server keeps statistical information for this zone, which can be dumped to the statistics file defined in the server options. See Section C.5.3.6.
sig-validity-interval	See the description of sig-validity-interval in Section C.5.3.6.13.
transfer-source	See the description of transfer-source in Section C.5.3.6.6.
transfer-source-v6	See the description of transfer-source-v6 in Section C.5.3.6.6.
notify-source	See the description of notify-source in Section C.5.3.6.6.
notify-source-v6	See the description of notify-source-v6 in Section C.5.3.6.6.
min-refresh-time max-refresh-time min-retry-time max-retry-time	See the descriptions of these options in Section C.5.3.6.13.

C.5.4 IPv6 Support in BIND Version 9

BIND supports all forms of IPv6 name-to-address and address-to-name lookups. It also uses IPv6 addresses to make queries when running on an IPv6-capable system.

The use of AAAA records is recommended because A6 records have been moved to experimental status. Like most stub resolvers, the resolver in TCP/IP Services supports only AAAA lookups because of the difficulty of following A6 chains.

For IPv6 lookups, the use of the nibble format used in the ip6.int domain is recommended because the bitstring format used in the ip6.arpa domain has been moved to experimental status. For more information about A6 and the bitstring format, refer to RFC 2874.

C.5.4.1 Address Lookups Using AAAA Records

The AAAA record is a parallel to the IPv4 A record. It specifies the entire address in a single record. For example:

```
$ORIGIN example.com.
host            3600    IN      AAAA    3ffe:8050:201:1860:42::1
```

C.5.4.2 Address-to-Name Lookups Using Nibble Format

As in IPv4, when looking up an address in nibble format, the address components are simply reversed and ip6.int. is appended to the resulting name. For example, the following would provide reverse name lookup for a host with address 3ffe:8050:201:1860:42::1:

```
$ORIGIN 0.6.8.1.1.0.2.0.0.5.0.8.e.f.f.3.ip6.int.
1.0.0.0.0.0.0.0.0.0.0.0.2.4.0.0   14400 IN      PTR     host.example.com.
```

C.5.5 DNS Notify

DNS Notify allows master name servers to notify their slave servers of changes to a zone's data. In response to a NOTIFY message from a master server, the slave checks to see whether its version of the zone is the current version. If it is not, the slave initiates a transfer. For more information, see the description of the notify option in Table C–7.

C.5.6 Incremental Zone Transfers (IXFR)

The incremental zone transfer (IXFR) protocol is a way for slave servers to transfer only changed data instead of having to transfer the entire zone. The IXFR protocol is documented in RFC 1995.

When acting as a master, BIND Version 9 supports IXFR for those zones in which the necessary change history information is available. These include master zones maintained by dynamic update and slave zones whose data was obtained by IXFR, but not manually maintained master zones and slave zones obtained by performing a full zone transfer (AXFR). When acting as a slave, BIND attempts to use IXFR unless it is explicitly disabled. For more information about disabling IXFR, see the description of the request-ixfr clause of the server statement in Section C.5.3.7.

C.5.7 Dynamic Updates

With dynamic updates, the BIND server can add, modify, or delete records or RRsets in the master zone files.

Dynamic updating is enabled on a zone-by-zone basis by including an allow-update or update-policy clause in the zone statement. Dynamic updating is described in RFC 2136.

Updating of secure zones (zones using DNSSEC) is presented in RFC 3007. SIG and NXT records affected by updates are automatically regenerated by the server using an online zone key. Update authorization is based on transaction signatures and on an explicit server policy.

C.5.7.1 The Journal File

All changes made to a zone using dynamic update are stored in the zone's journal file. This file is automatically created by the server when when the first dynamic update takes place. The name of the journal file is formed by appending _JNL to the name of the corresponding zone file. The journal file is in a binary format and should not be edited manually.

The server also occasionally writes (or "dumps") the complete contents of the updated zone to its zone file. This is not done immediately after each dynamic update; that would be too slow when a large zone is updated frequently. Instead, the dump is delayed by 15 minutes, allowing additional updates to take place.

When a server is restarted after a shutdown or failure, it replays the journal file to incorporate into the zone any updates that took place after the last zone dump. Changes that result from incoming incremental zone transfers are journaled in a similar way.

The zone files of dynamic zones normally cannot be edited because they are not guaranteed to contain the most recent dynamic changes—those are only in the journal file. The only way to ensure that the zone file of a dynamic zone is up to date is to use the `rndc stop` command.

If you have to make changes to a dynamic zone manually, use the following procedure:

1. Flush any dynamic updates in _JNL files to the master zone files using the following command:

   ```
   $ rndc flush-updates
   ```

2. Shut down the server using the following command:

   ```
   $ @SYS$STARTUP:TCPIP$BIND_SHUTDOWN.COM
   ```

3. Remove the zone's _JNL file.
4. Edit the zone file.
5. Restart the server using the following command:

   ```
   $ @ SYS$STARTUP:TCPIP_BIND_STARTUP.COM
   ```

Removing the _JNL file is necessary because the manual edits are not present in the journal, rendering it inconsistent with the contents of the zone file.

C.5.7.2 Dynamic Update Policies

BIND Version 9 supports two methods of granting clients the right to perform dynamic updates to a zone. You can configure them using either the `allow-update` option or the `update-policy` option.

The `allow-update` clause works the same way as in previous versions of BIND. It grants given clients the permission to update any record of any name in the zone.

The `update-policy` clause is new in BIND 9 and allows more fine-grained control over what updates are allowed. A set of rules is specified, where each rule either grants or denies permissions for one or more names to be updated by one or more identities. The rules apply to master zones only.

The update-policy statement only examines the signer of a message; the source address is not relevant. If the dynamic update request message is signed (that is, it includes either a TSIG or SIG(0) record), the identity of the signer can be determined.

If an allow-update statement appears when the update-policy statement is present, a configuration error occurs.

Use the following format to define rules:

(grant | deny) *identity nametype name* [*types*]

Each rule grants or denies privileges. Once a message has successfully matched a rule, the operation is immediately granted or denied and no further rules are examined. A rule is matched when the signer matches the identity field, the name matches the name field, and the type is specified in the type field. The rule definition includes the following fields:

- grant or deny specifies whether to grant or deny privileges.
- *identity* specifies the signer of the message. Use a name or a wildcard in the identity field.
- *name* specifies the name to be updated.
- *nametype* specifies one of the following:
  - *name*, which matches when the updated name is the same as the name in the name field.
  - subdomain, which matches when the updated name is a subdomain of the name in the name field (including the name itself).
  - wildcard, which matches an updated name that is a valid expansion of the wildcard name in the name field.
  - self, which matches when the updated name is the same as the message signer. The name field is ignored.
- *types* specifies the types of resource records.

  If no types are specified, the rule matches all types except SIG, NS, SOA, and NXT. Types can be specified by name, including ANY. ANY matches all types except NXT, which can never be updated.

C.5.7.3 Creating Updates Manually

If the name server for the domain is configured to accept dynamic updates, you can manually create updates to the domain database file using the nsupdate utility.

Note

Zones that are under dynamic control using nsupdate or a DHCP server should not be edited by hand. Manual edits could conflict with dynamic updates and could cause loss of data.

You start the utility using the NSUPDATE command, which is defined when you run the TCPIP$DEFINE_COMMANDS.COM procedure.

You can enter commands to the nsupdate utility interactively, or you can specify a file that contains nsupdate commands.

Transaction signatures can be used to authenticate update requests, as described in Section C.2.3. Signatures rely on a shared secret that should be known to only the nsupdate utility and the name server. TSIG uses the HMAC-MD5 encryption algorithm. It is important to specify the encryption algorithm because additional algorithms will be made available in the future. Use the appropriate configuration options in the server and key statements in TCPIP$BIND.CONF to ensure the name server associates the secret key and algorithm with the IP address of the client application that uses TSIG authentication. The nsupdate utility does not read the TCPIP$BIND.CONF file.

The format of the NSUPDATE command is:

```
NSUPDATE [-d] [-y keyname:secret | -k keyfile] [-v] [file-name]
```

In the NSUPDATE command line, you can include the following:

-d	Specifies debug mode.
-y *keyname:secret*	Generates a signature, where *keyname* specifies the name of the key and *secret* is a base-64 encoded secret. This option is not recommended because it displays the shared secret in plain text. Instead, use the -k option.
-k *keyfile*	Specifies a file (*keyfile*) that contains the shared secret. The file name has the following format: K*name*.157-*random*_PRIVATE For historical reasons, the file K*name*.157-*random*_KEY must also be present.
-v	Specifies that nsupdate use the TCP protocol instead of the UDP protocol. By default, nsupdate sends update requests using UDP.
file-name	Specifies a file that contains nsupdate commands.

If you do not specify the name of a command file, the NSUPDATE command prompts for a command line. The following list describes the commands for the nsupdate utility.

- server *servername* [*port*]

  Sends all dynamic update requests to the specified name server. When no name server is specified, the nsupdate utility sends updates to the master server of the correct zone. The MNAME field of that zone's SOA record identifies the master server for that zone. *port* is the port number to which the dynamic update requests are sent on the specified name server. If no port number is specified, the default DNS port number of 53 is used.

- local {*address*} [*port*]

  Sends all dynamic update requests using the local address. When no local address is provided, the nsupdate utility sends updates using an address and port chosen by the system. Specify *port* to make requests come from a specific port. If no port number is specified, the system assigns one.

- zone {*zonename*}

  Specifies that all updates are to be made to the specified zone. If no zone command is provided, the nsupdate utility attempts to determine the correct zone to update based on the rest of the input.

- key {*keyname secret*}

  Specifies that all updates are to be TSIG signed, using the specified keyname and key secret pair.

The key command overrides any key specified on the command line using the -y or -k options.

- prereq nxdomain *domain-name*

  Requires that no resource record of any type exist with the specified domain name.

- prereq yxrset *domain-name type* [*data*]

  Makes the presence of an RR set of the specified *type* owned by *domain-name* a prerequisite to performing the update. This requires that a resource record of the specified type, class, and domain name must exist. If *class* is omitted, IN (Internet) is assumed.

- prereq nxrrset {*domain-name*} [*class*] {*type*}

  Makes the nonexistence of an RRset of *type* owned by *domain-name* a prerequisite to performing the update specified in successive update commands. This requires that no resource record exist of the specified type, class, and domain name. If *class* is omitted, IN (Internet) is assumed.

  The data from each set of prerequisites of this form sharing a common type, class, and domain name are combined to form a set of resource records. This set of resource records must exactly match the set of resource records on the zone at the given type, class, and domain name. The data is written in the standard text representation of the resource record's RDATA.

- prereq yxdomain *domain-name*

  Makes the existence of the specified *domain-name* a prerequisite to performing the update. This requires that the domain name has at least one resource record of any type.

- update delete *domain-name ttl* [*class*] [*type*] [*rdata*]

  Deletes any resource records with the specified domain name. If *type* and *data* are provided, only matching resource records are removed. If *class* is omitted, IN (Internet) is assumed. The *ttl* value is ignored and is included only for compatibility.

- update add *domain-name ttl* [*class*] *type data*

  Adds a new resource record with the specified ttl, class, and data to the zone. The *ttl* value, the *type*, and the *data* must be included. The *class* is optional and defaults to IN.

- show

  Displays the current message, containing all of the prerequisites and updates specified since the last send command.

- send

  Sends the current message. This is equivalent to entering a blank line.

If you use a file to supply the updates, the data in the file must be in the following format:

class section name ttl type data

In this format:

- *class* is any one of the following keywords:
  - update

  - zone
  - prereq
- *section* is any one of the following keywords:
  - add
  - delete
  - nxdomain
  - yxdomain
  - nxrrset
  - yxrrset
- *name* is the name of the entry being added.
- *ttl* is the time to live (in seconds) for this entry. After this time period, the name server no longer serves the entry.
- *type* specifies the RR type (for example, A, CNAME, NS, MX, TXT).
- *rdata* specifies the data appropriate for the RR type being updated.

Lines beginning with a semicolon are comments and are ignored.

Examples

1. The following example shows how to supply a file (NSUPD.TXT) to the nsupdate utility.

```
$ TYPE NSUPD.TXT
update delete www.nads.zn.
update add www.nads.zn. 60 CNAME ivy18.nads.zn

$ NSUPDATE NSUPD.TXT
```

2. The following example shows how the nsupdate utility is used interactively to insert and delete resource records from the example.com zone. Notice that the input contains an extra blank line so that a group of commands are sent as one dynamic update request to the master name server for example.com.

```
$ NSUPDATE
> update delete oldhost.example.com A
> update add newhost.example.com 86400 A 172.16.1.1
>
```

   Any A records for oldhost.example.com are deleted, and an A record for newhost.example.com with IP address 172.16.1.1 is added. The newly added record has a TTL value of 86400 seconds (one day).

3. The following example tells the BIND server to verify the prerequisite condition that no resource records of any type exist for nickname.example.com. If any records exist, the update request fails. If no records with that name exist, a CNAME is added for it.

   This prerequisite condition is an RFC restriction that has been relaxed to allow for SIG, KEY, and NXT records to exist.

```
$ NSUPDATE
> prereq nxdomain nickname.example.com
> update add nickname.example.com CNAME somehost.example.com
>
```

After entering data in interactive mode, press Return (or Enter) on a line with no data to complete the input. The `nsupdate` utility then processes all update entries in one operation.

C.5.8 Configuring Cluster Failover and Redundancy

In the same OpenVMS Cluster, multiple BIND master servers can share a common database, thereby providing redundancy and a failover mechanism when one of the servers becomes unavailable.

To configure a DNS cluster failover and redundancy environment, perform the following steps on each node participating in the cluster.

1. Run the TCPIP$CONFIG command procedure, and from the Servers menu enable the BIND service.
2. Edit the BIND configuration file, SYS$SPECIFIC:[TCPIP$BIND]TCPIP$BIND.CONF.
   a. Configure the node as a master server.
   b. Add or edit the `options` statement. The `directory` substatement should be as follows:

```
options {
        directory "TCPIP$BIND_COMMON";
};
```

   TCPIP$BIND_COMMON is a logical name defined in the TCPIP$BIND_COMMON_STARTUP.COM command procedure as a search list. The search list consists of the SYS$SPECIFIC:[TCPIP$BIND] directory and the common directory. In the next step, the setup command procedure prompts you to specify the device on which the common directory is to reside. If you do not specify a device, the default device and directory is *common_device*:[TCPIP$BIND_COMMON], where *common_device* is generated automatically in the following manner:

   – If the SYSUAF logical is defined, the common disk is determined from its definition.
   – If the SYSUAF logical is not defined, the system uses SYS$SYSDEVICE as the default device.

3. Run the SYS$MANAGER:TCPIP$BIND_CLUSTER_SETUP.COM command procedure.

   This procedure creates two other command procedures that manage the startup and shutdown processes of the BIND component in a cluster environment:

   • SYS$MANAGER:TCPIP$BIND_COMMON_STARTUP.COM
   • SYS$MANAGER:TCPIP$BIND_COMMON_SHUTDOWN.COM

   These files define the BIND system logicals and accounting information. To remove the failover setup from your system, delete these two files.

4. Place any database files to be shared in the common directory.

Note

Be careful to remove from SYS$SPECIFIC:[BIND] any databases that are to be shared. Using the search list logical, BIND will find any

SYS$SPECIFIC:[BIND] databases first and use those. This might not be the result you want.

5. Start up BIND by entering the following command:

```
$ @SYS$MANAGER:TCPIP$BIND_STARTUP.COM
```

Caution

The use of dynamic updates in conjunction with a master BIND server that is participating in cluster failover and redundancy is not supported and might cause serious problems.

C.5.8.1 Changing the BIND Database

If multiple master BIND servers are running in a cluster, and a change is made to the common BIND database, the database must be reloaded on each node that is running the master BIND server. To reload the BIND database on every node in the cluster where the master BIND server is running, enter the following command:

```
TCPIP> SET NAME_SERVICE /INITIALIZE /CLUSTER=dev:[directory]
```

The /CLUSTER qualifier takes the directory specification of the common BIND directory as a value. If you omit the device and directory, they default to:

```
common_device:[TCPIP$BIND_COMMON]
```

In this case, *common_device* is generated automatically in the following manner:

- If the SYSUAF logical is defined, the the common disk is determined from its definition.
- If SYSUAF logical is not defined, the system uses SYS$SYSDEVICE as the default device.

C.6 Populating the BIND Server Databases

To populate the BIND server database files, use one of the following methods:

- Convert an existing host database with the CONVERT/UNIX BIND command.
- Manually edit the ZONE.DB files.

C.6.1 Using Existing Databases

To populate the BIND server database by copying information from the hosts database and other database files, enter the CONVERT/UNIX BIND command. This command:

- Creates a BIND server database (if needed).
- Extracts data from the hosts database. (The BIND server uses UNIX style formatted files.)
- Extracts Mail Exchange (MX) information from the routes database.
- Populates the BIND server database with the host and MX records.

- Creates a forward translation file with the following characteristics:
  - It has address, canonical name, and MX entries.
  - If a file with the same name as the output file already exists, the serial number from that file's start-of-authority (SOA) entry increments and becomes the serial number of the new output file.
  - If no previous version of the output file exists, the serial number for the new file is 1.

  When you specify forward translation (by omitting the /DOMAIN qualifier), any host in the hosts database that is not qualified with a domain is included in the target domain. For example, if the local domain is `x.y.z.`, the CONVERT/UNIX BIND command includes: `a`, `b.x.y.z`, `c.x.y.z.z` but does not include `d.x.y.h`.
- Creates a reverse translation file if you specify /DOMAIN=(*domain.name*) and the end of *domain.name* is IN-ADDR.ARPA.

  The created reverse translation file has the following characteristics:
  - Only records applicable to the domain you specify are placed into the output file.
  - The output file has domain name pointer entries.
  - If a file with the same name as the output file already exists, the serial number from that file's SOA entry increments and becomes the serial number of the new output file.
  - If no previous version of the output file exists, the serial number for the new file is 1.
  - The file selects hosts with IP addresses that match the partial IP address from *domain.name*. For example, /DOMAIN=16.99.IN-ADDR.ARPA does a reverse translation and selects hosts whose addresses begin with 99.16.

If the BIND server's directory is SYS$SPECIFIC:[TCPIP$BIND] and you have specified domain `abc.def.com`, the default output file is named SYS$SPECIFIC:[TCPIP$BIND]ABC_DEF_COM.DB.

Compaq suggests that you do not change the default directory name. If you do, the file is created in your current directory.

On the command line, specify the full OpenVMS file specification. Do not specify a version number, and do not use wildcards. The following example uses the domain `ucx.ern.sea.com`, creates a UCX_ERN_SEA_COM.DB file, creates a 208_20_9_IN-ADDR_ARPA.DB file, and checks the results by displaying directory listings with the new file.

```
TCPIP> CONVERT/UNIX BIND /DOMAIN=UCX.ERN.SEA.COM
TCPIP> CONVERT/UNIX BIND /DOMAIN=208.20.9.IN-ADDR.ARPA

TCPIP> SET DEFAULT SYS$SPECIFIC:[TCPIP$BIND]
$ DIRECTORY

Directory SYS$SPECIFIC:[TCPIP$BIND]

127_0_0.DB;1        208_20_9_IN-ADDR_ARPA.DB;1
LOCALHOST.DB;1
LOGIN.COM;1         ROOT.HINT;1         TCPIP$BIND.CONF;1
TCPIP$BIND_CONF.TEMPLATE;1              TCPIP$BIND_RUN.LOG;4339
TCPIP$BIND_SERVER.PID;1                 UCX_ERN_SEA_COM.DB;5
```

C.6.2 Manually Editing Zone Files

All name server zone files use the same type of records to define domain database information. Compaq recommends that you review these resource records before you edit any BIND files. Table C–21 describes the standard resource records (RR).

Table C–21 Standard Resource Record Types

Record Type	Description
SOA	Start of authority. Marks the beginning of a zone's data and defines parameters that affect the entire zone.
NS	Name server. Identifies a domain's name server.
A	Address. Maps a host name to an address.
PTR	Pointer. Maps an address to a host name.
MX	Mail Exchange. Identifies where to deliver mail for a given domain.
CNAME	Canonical name. Defines an alias host name.
HINFO	Host information. Describes a host's hardware and operating system.
WKS	Well-known service. Advertises network services.

The format of DNS records is as follows:

```
[name] [ttl] IN type data
```

In this format:

name	Specifies the name of the domain object referenced by a resource record. The string entered for *name* is the current domain unless it ends with a dot. If the name field is blank, the record applies to the domain object last named.
ttl	Defines the length of time, in seconds, that the information in this resource record should be kept in cache. Usually, the time-to-live field is left blank, and the default *ttl*, set for the entire zone SOA record, is used.
IN	Identifies the record as an Internet DNS resource record.
type	Identifies what kind of resource record this is. (See Table C–21 for the record types you can specify.)
data	Information specific to this type of resource record. For example, in an A record, this is the field that contains the actual IP address.

C.6.2.1 Setting TTLs

The time to live (TTL) of the RR field is a 32-bit integer that represents the number of seconds that an RR can be cached before it should be discarded. The following types of TTL values are used in a zone file:

- SOA

  The last field in the SOA is the negative caching TTL. This controls how long other servers cache no-such-domain (NXDOMAIN) responses from you.

  The maximum time for negative caching is 3 hours (3h).

- $TTL

  The $TTL directive at the top of the zone file (before the SOA) gives a default TTL for every RR without a specific TTL set.

- RR TTLs

  Each RR can have a TTL as the second field in the RR, which controls how long other servers can cache it.

All of these TTLs default to units of seconds, though units can be explicitly specified (for example, `1h30m` for 1 hour and 30 minutes).

C.6.2.2 Zone File Directives

While the master file format itself is class independent, all records in a master file must be of the same class. The master file directives are described in the following list:

- `$ORIGIN` *domain-name* [*comment*]

  Sets the domain name that is appended to any unqualified records. When a zone is first read, an implicit $ORIGIN *zone-name* directive is applied.

  If domain specified is not absolute, the current $ORIGIN is appended to it.

  For example, the following are interpreted the same way:

```
$ORIGIN example.com
WWW     CNAME   MAIN-SERVER
```

  And:

```
WWW.EXAMPLE.COM. CNAME MAIN-SERVER.EXAMPLE.COM.
```

- `$INCLUDE` *filename* [*origin*] [*comment*]

  Reads and processes the specified file as if it were included into the file at this point. If *origin* is specified, the file is processed with $ORIGIN set to that value; otherwise, the current $ORIGIN is used.

  Once the file has been read, the origin and the current domain name revert to the values they had prior to the $INCLUDE.

- `$TTL` *default-ttl* [*comment*]

  Sets the default time to live (TTL) for subsequent records with undefined TTLs. Valid TTLs are in the range of 0—2147483647 seconds.

C.6.3 Saving Backup Copies of Zone Data

The name server saves backup copies of the zone data in SYS$SPECIFIC:[TCPIP$BIND]. Do not delete these backup copies. When the master server is down and the secondary server is started, the secondary server cannot perform a zone transfer until the master server is up. However, with backup copies, the secondary server has some data (though possibly out of date) to perform its basic tasks.

C.6.4 Sample Database Files

The following sections provide sample BIND database files.

C.6.4.1 Local Loopback

In the LOCALHOST.DB file, the local host address is usually 127.0.0.1. The following sample LOCALHOST.DB file shows the forward translation for the local loopback interface:

```
;
; BIND data file for local loopback interface (forward
translation).
;
; Provided for Compaq TCP/IP Services for OpenVMS.
;
$ORIGIN localhost.
@                1D IN SOA       @ root (
                                 42              ;Serial
                                 3H              ;Refresh
                                 15M             ;Retry
                                 1W              ;Expiry
                                 1D )            ;Minimum
;
                 1D IN NS        @
                 1D IN A         127.0.0.1
```

The following sample 127_0_0.DB file shows the reverse translation for the local loopback interface:

```
;
; BIND data file for local loopback interface (reverse
translation).
;
; Provided for Compaq TCP/IP Services for OpenVMS.
;
$ORIGIN 0.0.127.in-addr.arpa.
@                1D IN SOA       localhost.
root.localhost. (
                                 42              ;Serial
                                 3H              ;Refresh
                                 15M             ;Retry
                                 1W              ;Expiry
                                 1D )            ;Minimum
;
                 1D IN NS        localhost.
1                1D IN PTR       localhost.
```

These local host databases provide forward and reverse translation for the widely used LOCALHOST name. The LOCALHOST name is always associated with the IP address 127.0.0.1 and is used for local loopback traffic.

C.6.4.2 Hint File

This file contains root name server hints. Any name server running on a host without direct Internet connectivity should list the internal roots in its hint file.

The following sample shows a ROOT.HINT file. In earlier releases, this file was called NAMED.CA:

```
; Data file for initial cache data for root domain servers.
;
; Provided for Compaq TCP/IP Services for OpenVMS.
;
; <<>> DiG 9.2.0 <<>>
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 11672
;; flags: qr rd ra; QUERY: 1, ANSWER: 13, AUTHORITY: 0, ADDITIONAL: 13

;; QUESTION SECTION:
;.                               IN      NS
```

```
;; ANSWER SECTION:
.                       417496  IN      NS      E.ROOT-SERVERS.NET.
.                       417496  IN      NS      F.ROOT-SERVERS.NET.
.                       417496  IN      NS      G.ROOT-SERVERS.NET.
.                       417496  IN      NS      H.ROOT-SERVERS.NET.
.                       417496  IN      NS      I.ROOT-SERVERS.NET.
.                       417496  IN      NS      J.ROOT-SERVERS.NET.
.                       417496  IN      NS      K.ROOT-SERVERS.NET.
.                       417496  IN      NS      L.ROOT-SERVERS.NET.
.                       417496  IN      NS      M.ROOT-SERVERS.NET.
.                       417496  IN      NS      A.ROOT-SERVERS.NET.
.                       417496  IN      NS      B.ROOT-SERVERS.NET.
.                       417496  IN      NS      C.ROOT-SERVERS.NET.
.                       417496  IN      NS      D.ROOT-SERVERS.NET.

;; ADDITIONAL SECTION:
A.ROOT-SERVERS.NET.     503896  IN      A       198.41.0.4
B.ROOT-SERVERS.NET.     503896  IN      A       128.9.0.107
C.ROOT-SERVERS.NET.     503896  IN      A       192.33.4.12
D.ROOT-SERVERS.NET.     503896  IN      A       128.8.10.90
E.ROOT-SERVERS.NET.     503896  IN      A       192.203.230.10
F.ROOT-SERVERS.NET.     503896  IN      A       192.5.5.241
G.ROOT-SERVERS.NET.     503896  IN      A       192.112.36.4
H.ROOT-SERVERS.NET.     503896  IN      A       128.63.2.53
I.ROOT-SERVERS.NET.     503896  IN      A       192.36.148.17
J.ROOT-SERVERS.NET.     503896  IN      A       198.41.0.10
K.ROOT-SERVERS.NET.     503896  IN      A       193.0.14.129
L.ROOT-SERVERS.NET.     503896  IN      A       198.32.64.12
M.ROOT-SERVERS.NET.     503896  IN      A       202.12.27.33

;; Query time: 2144 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Wed Oct 17 12:03:16 2001
;; MSG SIZE  rcvd: 436
```

This cache initialization file contains NS records that name root servers and A records that provide the addresses of root servers.

To create a ROOT.HINT file:

1. Run TCPIP$CONFIG.
2. Select the Server Components menu.
3. Select the BIND server.
4. Enable the BIND server.

This procedure creates the ROOT.HINT file and places the file in the SYS$SPECIFIC:[TCPIP$BIND] directory.

C.6.4.3 Forward Translation File

The forward translation file, *domain_name*.DB, stores host-name-to-address mapping. For example, the database file ROBIN_BIRD_COM.DB is created for the domain ROBIN.BIRD.COM.

The following example shows a *domain_name*.DB file:

```
$TTL 86400
$ORIGIN ucx.ern.sea.com.
@               IN      SOA     owl.ucx.ern.sea.com. pmaster.owl.ern.sea.com.
(
                                23      ; Serial
                                600     ; Refresh
                                300     ; Retry
                                172800  ; Expire
                                43200 ) ; Minimum
;
                IN      NS      owl.ucx.ern.sea.com.
                IN      NS      condor.ucx.ern.sea.com.
;
thrush          IN      A       9.20.208.53
condor          IN      A       9.20.208 or 90
birdy           IN      A       9.20.208.47
                IN      MX      10 birdy.ucx.ern.sea.com.
                IN      MX      100 inet-gw-1.pa.emu.com.
                IN      MX      100 mts-gw.pa.emu.com.
                IN      MX      200 crl.emu.com.
                IN      MX      300 nester.emu.com.
seagull         IN      A       9.20.208.30
                IN      MX      10 seagull.ucx.ern.sea.com.
                IN      MX      100 inet-gw-1.pa.emu.com.
                IN      MX      100 mts-gw.pa.emu.com.
                IN      MX      200 crl.emu.com.
                IN      MX      300 nester.emu.com.
owl             IN      A       9.20.208.72
                IN      MX      10 owl.ucx.ern.sea.com.
                IN      MX      100 inet-gw-1.pa.emu.com.
                IN      MX      100 mts-gw.pa.emu.com.
                IN      MX      200 crl.emu.com.
                IN      MX      300 nester.emu.com.
peacock         IN      A       9.20.208.73
                IN      MX      10 pultdown.ucx.ern.sea.com.
                IN      MX      100 inet-gw-1.pa.emu.com.
                IN      MX      100 mts-gw.pa.emu.com.
                IN      MX      200 crl.emu.com.
                IN      MX      300 nester.emu.com.
redwing         IN      A       9.20.208.79
                IN      MX      10 redwing.ucx.ern.sea.com.
                IN      MX      100 inet-gw-1.pa.emu.com.
                IN      MX      100 mts-gw.pa.emu.com.
                IN      MX      200 crl.emu.com.
                IN      MX      300 nester.emu.com.
robin           IN      A       9.20.208.47
                IN      A       9.20.208.30
                IN      A       9.20.208.72
```

This file is created only for the master server. All other servers obtain this information from the master server. This file contains most of the domain information and has the following characteristics:

- Begins with an SOA record and a few NS records that define the domain and its servers.
- Maps host names to IP addresses.
- Contains A, MX, CNAME, and other records.

MX records identify the servers in a domain that are used for forwarding mail. Use MX records and preference numbers to define the order in which mail servers are used. The lower the preference number, the more desirable the server.

C.6.4.4 Reverse Translation File

The reverse translation file, *address*.DB, stores address-to-host-name mapping (reverse mapping) information. For example, for the same domain, a file with the name 208_20_9_IN-ADDR_ARPA.DB is created.

The following example shows an *address*.DB file:

```
$TTL 86400
$ORIGIN 208.20.9.in-addr.arpa.
@     IN   SOA   owl.ucx.ern.sea.com. pmaster.owl.ucx.ern.sea.com.
(
                          1       ; Serial
                          600     ; Refresh
                          300     ; Retry
                          172800  ; Expire
                          43200 ) ; Minimum
;
      IN      NS      owl.ucx.ern.sea.com.
      IN      NS      condor.ucx.ern.sea.com.
;
53              IN      PTR     thrush.ucx.ern.sea.com.
10              IN      PTR     condor.ucx.ern.sea.com.
47              IN      PTR     birdy.ucx.ern.sea.com.
30              IN      PTR     seagull.ucx.ern.sea.com.
72              IN      PTR     owl.ucx.ern.sea.com.
73              IN      PTR     peacock.ucx.ern.sea.com.
79              IN      PTR     redwing.ucx.ern.sea.com.
```

PTR records predominate in this file because they are used to translate addresses to host names.

C.7 Examining Name Server Statistics

The BIND server collects statistics that record server activity. To examine BIND statistics, use one of the following commands:

- The TCP/IP management command SHOW NAME_SERVICE/STATISTICS
- The `rndc stats` command

Statistics are logged to the TCPIP$BIND.STATS file, located in SYS$SPECIFIC:[TCPIP$BIND].

The following sample shows a statistics log:

```
+++ Statistics Dump +++ (1004986341)
success 17
referral 0
nxrrset 1
nxdomain 1
recursion 6
failure 0
--- Statistics Dump --- (1004986341)
```

The statistics dump begins with the line `+++ Statistics Dump +++ (973798949)`. The number in parentheses is a standard UNIX timestamp, measured as seconds since January 1, 1970. Following that line are a series of lines containing a counter type, the value of the counter, a zone name (optional), and a view name (optional).

The lines without view and zone listed are global statistics for the entire server. Lines with a zone and view name are for the given view and zone. (The view name is omitted for the default view.)

The statistics dump ends with the line `--- Statistics Dump --- (973798949)` The number in parentheses is identical to the number in the beginning line.

The following statistics counters are maintained:

- `success`

  The number of successful queries made to the server or zone. A successful query is defined as query that returns a NOERROR response other than a referral response.

- `referral`

  The number of queries that resulted in referral responses.

- `nxrrset`

  The number of queries that resulted in NOERROR responses with no data.

- `nxdomain`

  The number of queries that resulted in NXDOMAIN responses.

- `recursion`

  The number of queries that caused the server to perform recursion in order to find the final answer.

- `failure`

  The number of queries that resulted in a failure response other than those described in the previous counters.

C.8 Configuring BIND with the SET CONFIGURATION Command

The following sections describe how to set up BIND servers manually using the TCP/IP management command SET CONFIGURATION BIND.

Note

This command creates a UCX Version 4.*x* configuration. If you set up your BIND name server using this command, you must also use the TCP/IP management command CONVERT/CONFIGURATION BIND command to convert the databases to the BIND Version 9 format. If you omit this step, your changes will not take effect.

C.8.1 Setting Up a Master Name Server

To instruct the master name server to read the appropriate database files using the information in TCPIP$CONFIGURATION.DAT, use the SET CONFIGURATION BIND command. Use the SHOW CONFIGURATION BIND command to display BIND information from the configuration database (TCPIP$CONFIGURATION.DAT).

The following commands tell the name server to read the appropriate files:

```
TCPIP> SET CONFIGURATION BIND /CACHE

TCPIP> SET CONFIGURATION BIND -
_TCPIP> /PRIMARY=(DOMAIN:0.0.127.IN-ADDR.ARPA, FILE:NAMED.LOCAL)

TCPIP> SET CONFIGURATION BIND -
_TCPIP> /PRIMARY=(DOMAIN:UCX.ERN.SEA.COM, FILE:UCX_ERN_SEA_COM.DB)
```

```
TCPIP> SET CONFIGURATION BIND -
_TCPIP> /PRIMARY=(DOMAIN:208.20.9.IN-ADDR.ARPA, FILE:208_20_9_IN-ADDR_ARPA.DB)
```

To view these settings, use the SHOW CONFIGURATION BIND command.

C.8.2 Setting Up a Secondary (Slave) Name Server

You can configure a secondary server to populate itself by copying the DNS database files from the master server.

To configure a secondary server, enter the following commands:

```
TCPIP> SET CONFIGURATION BIND /CACHE

TCPIP> SET CONFIGURATION BIND -
_TCPIP> /PRIMARY=(DOMAIN:0.0.127.IN-ADDR.ARPA, FILE:NAMED.LOCAL)

TCPIP> SET CONFIGURATION BIND -
_TCPIP> /SECONDARY=(DOMAIN:UCX.ERN.SEA.COM, -
_TCPIP> FILE:UCX_ERN_SEA_COM.DB,HOST:OWL)

TCPIP> SET CONFIGURATION BIND -
_TCPIP> /SECONDARY=(DOMAIN:208.20.9.IN-ADDR.ARPA, -
_TCPIP> FILE:208_20_9_IN-ADDR_ARPA.DB, -
_TCPIP> HOST:OWL.UCX.ERN.SEA.COM)
```

C.8.3 Setting Up a Cache-Only Server

To configure a cache-only server, enter the following command:

```
TCPIP> SET CONFIGURATION BIND /CACHE
```

This command points the server to the file NAMED.CA.

C.8.4 Setting Up a Forwarder Name Server

To configure a forwarder server, enter the following command:

```
TCPIP> SET CONFIGURATION BIND /FORWARDERS=(HOST:host)
```

In this command, *host* specifies the forwarding server.

Note

You cannot set up a server to be both a forwarder and a caching server.

C.9 Configuring the BIND Resolver

Your host uses the BIND resolver to obtain information from a name server. When a request for name translation arrives, the resolver first searches the local host database for the host information. If the information is not found, the resolver then queries the BIND name server for host information.

Note

The BIND resolver is based on the BIND Version 8 implementation of DNS.

The resolver is automatically configured by TCPIP$CONFIG when you choose Option 1 --- Core Environment. To display your resolver configuration, enter the following TCP/IP management command:

```
TCPIP> SHOW NAME_SERVICE
```

TCP/IP Services displays the following data:

```
BIND Resolver Parameters

 Local domain: ucx.ern.sea.com

 System

  State:     Started, Enabled

  Transport: UDP
  Domain:    ucx.ern.sea.com
  Retry:     4
  Timeout:   4
  Servers:   lark
  Path:      ucx.ern.sea.com,ern.sea.com,sea.com

 Process

  State:     Enabled

  Transport:
  Domain:
  Retry:
  Timeout:
  Servers:
  Path:
```

Here, host LARK in the current domain is the default name server. To add records to the local hosts database, use the SET HOST command. For example, the following command adds host birdy to the local hosts database. (For more information about using SET commands, see the *Compaq TCP/IP Services for OpenVMS Management Command Reference* manual.)

```
TCPIP> SET HOST birdy /ADDRESS=9.20.208.47
```

To delete server entries from the configuration database or to add new entries, enter the following command:

```
TCPIP> SET NAME_SERVICE /NOSERVER=LARK /SYSTEM
```

This command modifies the volatile database. To the the change to the permanent database, enter the SET CONFIGURATION NAME_SERVICE command.

To view the results, enter the SHOW CONFIGURATION NAME_SERVICE command.

C.9.1 Changing the Default Configuration

To add a new server and enable the BIND resolver, enter the following command:

```
TCPIP> SET NAME_SERVICE /SERVER=host /ENABLE /SYSTEM
```

For *host*, specify the host name or IP address of the BIND server or servers that the BIND resolver is to query.

To specify multiple hosts, list them by request preference. The BIND resolver sends the first lookup request to the first host on the list.

If you define a server list and then add a new server with the SET NAME_SERVICE /SERVER command, the new server is added to the end of the list.

SET commands affect the volatile database. To save your changes to the permanent database, use the SET CONFIGURATION commands. The changes you make with the SET CONFIGURATION commands take effect the next time the software starts up. For example:

```
TCPIP> SET CONFIGURATION NAME_SERVICE /SERVER=host /ENABLE

TCPIP> SHOW CONFIGURATION NAME_SERVICE

BIND Resolver Configuration

  Transport:  UDP
  Domain:     ucx.ern.sea.com
  Retry:         4
  Timeout:       4
  Servers:    9.20.208.47, 9.20.208.53
  Path:       No values defined
```

C.9.2 Examples

The following command defines hosts PARROT, SORA, and JACANA as systemwide BIND servers and enables the BIND resolver:

```
PARROT> TCPIP
TCPIP> SET NAME_SERVICE /SERVER=(PARROT,SORA,JACANA) /SYSTEM /ENABLE
```

The following example defines, for the current login session, host OSPREY as the BIND server. As a result, the servers that are defined systemwide are not queried.

```
TCPIP> SET NAME_SERVICE /SERVER=OSPREY
```

C.9.3 Resolver Default Search Behavior

By default, if no search list is defined and the host name as you typed it has no dot (.) in the name, the BIND resolver performs a lookup using the following forms of the host name (in this order):

1. The host name, with the default domain appended
2. Just the host name

For example, suppose you enter the following command:

```
TCPIP> SHOW HOST OWL
```

Assuming that the default domain is `ucx.ern.sea.com`, the resolver performs lookups as follows:

1. On the host name and domain `owl.ucx.ern.sea.com`.
2. If that lookup was unsuccessful, the resolver searches for host `owl`.

This behavior is different than the resolver lookup behavior in previous releases (UCX BIND Version 4.*x*.). The following section provides more information.

C.9.4 Resolver Search Behavior in Earlier Releases

In previous releases, the resolver performed lookups as follows:

1. Appended the default domain to the host name and performed a lookup.
2. If the previous lookup failed, the resolver removed the leftmost label from the default domain name, appended the result to the host name and performed the lookup.
3. If that lookup failed, the resolver again removed the leftmost label from the default domain name, appended the result to the host name, and performed the lookup.

For each unsuccessful lookup, this procedure was repeated until only two labels remained in the resulting domain name.

If all these attempts failed, the resolver tried just the host name as typed (as long as it contained at least one dot).

For example, suppose you entered the following command:

```
TCPIP> SHOW HOST OWL
```

Assuming the default domain was `ucx.ern.sea.com`, the resolver performed lookups as follows:

1. On `owl.ucx.ern.sea.com`.
2. If the previous lookup was unsuccessful, the resolver searched for `owl.ern.sea.com`.
3. If that lookup was unsuccessful, the resolver searched for `owl.sea.com`.
4. Finally, if the preceding lookup was unsuccessful, the resolver searched for `owl`.

C.9.5 Setting the Resolver's Domain Search List

The search list is provided to make entering lookup commands easier by not requiring you to type fully qualified domain names. The search list consists of domain names that the resolver uses when performing lookups. By default, the search list consists of only the default domain, which is stored in the TCPIP$CONFIGURATION.DAT file.

You can change the elements in the search list by entering the SET NAME_SERVICE command, as shown in the following example:

```
TCPIP> SET NAME_SERVICE /PATH=(ucx.ern.sea.com,dux.sea.com,mux.ern.sea.com)/SYSTEM
```

For example, suppose you enter the following command:

```
TCPIP> SHOW HOST CANARY
```

The resolver performs lookups as follows:

1. On `canary.ucx.ern.sea.com`.
2. If the previous lookup was unsuccessful, the resolver searches for `canary.dux.sea.com`.
3. If that lookup was unsuccessful, the resolver searches for `canary.mux.ern.sea.com`.
4. If that lookup was unsuccessful, the resolver searches for `canary`.

In the following output of the SHOW NAME_SERVICE command, the PATH: label shows the search list information entered with the SET NAME_SERVICE /PATH command. This command displays systemwide information and process-specific information (if process-specific information is set).

```
TCPIP> SHOW NAME_SERVICE

BIND Resolver Parameters

Local domain: ucx.ern.sea.com

System

State:     Started, Enabled

Transport: UDP
Domain:    ucx.ern.sea.com
Retry:     4
Timeout:   4
Servers:   ucx, lemng, 16.99.0.10
Path:      ucx.ern.sea.com, dux.ern.sea.com, mux.ern.sea.com

Process

State:     Enabled
Transport:
Domain:
Retry:
Timeout:
Servers:
Path:
$
```

Any additions you make are appended to the end of the search list.

To remove an element from the search list, enter the following command:

```
TCPIP> SET NAME_SERVICE /NOPATH=dux.ern.sea.com /SYSTEM
```

Note

When you run TCPIP$CONFIG.COM after upgrading from UCX to TCP/IP Services for OpenVMS, the system creates a domain search list that is consistent with the UCX default lookup behavior. TCPIP$CONFIG.COM uses the default domain to create a search list consisting of each parent domain. For example, if the default domain is ucx.ern.sea.com, the resulting search list is ucx.ern.sea.com,ern.sea.com,sea.com. You can modify the current search list by using the SET CONFIGURATION NAME_SERVER /PATH command.

C.10 BIND Server Administrative Tools

The following administrative tools play an integral part in the management of a server.

- The bind_checkconf utility checks the syntax of the BIND server configuration file.
- The bind_checkzone utility checks a zone file for syntax and consistency.
- The dnssec_keygen generates keys for DNSSEC (secure DNS) and TSIG (transaction signatures).

- The dnssec_makekeyset utility generates a key set.
- The dnssec_signkey utility signs a key set.
- The dnssec_signzone utility signs a zone.
- The rndc utility allows you to control the operation of a name server.
- The rndc_confgen utility generates configuration files for the rndc utility.

To use these utilities, you must have system management privileges. Run the TCPIP$DEFINE_COMMANDS.COM procedure to define the commands described in the following reference sections.

bind_checkconf

Checks the syntax of a BIND server configuration file.

Format

bind_checkconf [-v] [-t *directory*] *filename*

Description

The bind_checkconf utility checks the syntax, but not the semantics, of a BIND server configuration file.

Options

-t *directory*
Looks for *filename* in the specified directory. The default directory is SYS$SPECIFIC:[TCPIP$BIND].

-v
Displays only the version number of the bind_checkconf utility and exits.

filename
Specifies the name of the configuration file to be checked. The default file is SYS$SPECIFIC:[TCPIP$BIND]TCPIP$BIND.CONF.

bind_checkzone

Checks a zone file for syntax and consistency.

Format

bind_checkzone [-d] [-q] [-v] [-c *class*] [-t *directory*] *zonename filename*

Description

The bind_checkzone utility checks the syntax and integrity of a zone file. It performs the same checks as the BIND server does when it loads a zone. This makes bind_checkzone useful for checking zone files before configuring them into a name server.

Options

-d
Enables debugging mode.

-q
Enables quiet mode (exit code only).

-v
Print the version number of bind_checkzone and exits.

-c *class*
Specifies the class of the zone. If not specified, the default is IN.

-t *directory*
Looks for the zone in the specified directory. The default directory is SYS$SYSPECIFIC:[TCPIP$BIND].

zonename
Specifies the name of the zone being checked.

filename
Specifies the name of the zone file.

dnssec_keygen

Generates keys for DNSSEC.

Format

dnssec_keygen -a *algorithm* -b *keysize* -n *nametype* [-c *class*] [-e] [-g *generator*] [-h] [-p *protocol*] [-r *randomfile*] [-t *type*] [-v *level*] *name*

Description

The dnssec_keygen generates keys for DNSSEC, as defined in RFC 2535. It can also generate keys for use with TSIG (Transaction Signatures), as defined in RFC 2845.

Parameters

-a *algorithm*
Selects the cryptographic algorithm. The value of *algorithm* must be one of the following:

- RSAMD5
- RSA
- DSA
- DH (Diffie-Hellman)
- HMAC-MD5

These values are not case sensitive.

-b *keysize*
Specifies the number of bits in the key. The choice of key size depends on the algorithm used:

- RSA keys must be between 512 and 4096 bits.
- Diffie-Hellman keys must be between 128 and 4096 bits.
- DSA keys must be between 512 and 1024 bits and must be an exact multiple of 64.
- HMAC-MD5 keys must be between 1 and 512 bits.

-n *nametype*
Specifies the owner type of the key. The value of *nametype* must one of the following:

- ZONE (for a DNSSEC zone key)
- HOST or ENTITY (for a key associated with a host)
- USER (for a key associated with a user)

These values are not case sensitive.

name
Specifies the name of the domain.

Options

-c *class*
Indicates that the DNS record containing the key should have the specified class. If not specified, class IN is used.

-e
If generating an RSA key, specifies the use of a large exponent.

-g *generator*
If generating a Diffie-Hellman key, specifies the generator. Allowed values for *generator* are 2 and 5. If no generator is specified, a known prime from RFC 2539 is used, if possible; otherwise the default is 2.

-h
Displays a short summary of the options and arguments to the `dnssec_keygen` command.

-p *protocol*
Sets the protocol value for the generated key. The value of *protocol* is a number between 0 and 255. For keys of type USER, the default is 2 (e-mail). For all other key types, the default is 3 (DNSSEC). Other possible values for this argument are listed in RFC 2535 and its successors.

-r *randomfile*
Specifies the source of randomness. The default source of randomness is keyboard input. *randomfile* specifies the name of a file containing random data to be used instead of the default. The special value `keyboard` indicates that keyboard input should be used.

Note

When you use the keyboard to generate random data, you must input a large amount of data. Input requiring hundreds of lines of data is not unusual for some algorithms. The string "stop typing" appears when enough data has been input.

-s *strength*
Specifies the strength value of the key. The value of *strength* is a number between 0 and 15. This option is currently not used.

-t *type*
Indicates the use of the key. The *type* must be one of the following:

- AUTHCONF (authenticate and encrypt data)
- NOAUTHCONF (do not authenticate and do not encrypt data)
- NOAUTH (do not authenticate data)
- NOCONF (do not encrypt data)

The default is AUTHCONF.

-v *level*
Sets the debugging level.

Generated Keys

When dnssec_keygen completes successfully, it displays a string of the following form to standard output:

Knnnn.aaa-iiiii

This is an identification string for the key it has generated. These strings can be used as arguments to the dnssec_makekeyset utility. The string is interpreted as follows:

- *nnnn* is the key name.
- *aaa* is the numeric representation of the algorithm.
- *iiiii* is the key identifier (or footprint).

dnssec_keygen creates two files, with names based on the printed string. The file K*nnnn.aaa-iiiii*_KEY contains the public key, and K*nnnn.aaa-iiiii*_PRIVATE contains the private key.

The _KEY file contains a DNS KEY record that can be inserted into a zone file (either directly, or using an $INCLUDE statement).

The _PRIVATE file contains algorithm-specific fields. For security reasons, this file does not have general read permission.

Both _KEY and _PRIVATE files are generated for symmetric encryption algorithms such as HMAC-MD5, even though the public and private key are equivalent.

Examples

To generate a 768-bit DSA key for the domain example.com, enter the following command:

1. `$ dnssec_keygen -a DSA -b 768 -n ZONE example.com`

   This command displays a string of the form:

   `Kexample_com.003-26160`

   In this example, dnssec_keygen creates the files KEXAMPLE_COM.003-26160_KEY and KEXAMPLE_COM.003-26160_PRIVATE.

dnssec_makekeyset

Generates signed key sets for DNSSEC.

Format

dnssec_makekeyset [-a] [-s *start-time*] [-e *end-time*] [-h] [-p] [-r *randomfile*] [-t *ttl*] [-v *level*] *key*...

Description

The dnssec_makekeyset utility generates a key set from one or more keys created by the dnssec_keygen utility. It creates a file containing a KEY record for each key, and self-signs the key set with each zone key. The output file is of the form KEYSET-*name*.DAT, where *name* is the zone name.

Options

-a
Verifies all generated signatures.

-s *start-time*
Specifies the date and time when the generated SIG records become valid. This can be either an absolute or relative time. An absolute start time is indicated by a number in *YYYYMMDDHHMMSS* notation. 20000530144500 denotes 14:45:00 UTC on May 30, 2000. A relative start time is indicated by *+N*, which is *N* seconds from the current time. If no start time is specified, the current time is used.

-e *end-time*
Specifies the date and time when the generated SIG records expire. An absolute end time is indicated in *YYYYMMDDHHMMSS* notation. A time relative to the start time is indicated by *+N*, which is *N* seconds from the start time. A time relative to the current time is indicated by now+*N*. If no end time is specified, 30 days from the start time is used as a default.

-h
Displays a short summary of the options and arguments to the dnssec_makekeyset command.

-p
Uses pseudorandom data when signing the zone. This is faster, but less secure, than using real random data. This option is useful when signing large zones or when the entropy source is limited.

-r *randomfile*
Specifies the source of randomness. The default source of randomness is keyboard input. The argument *randomfile* specifies the name of a file containing random data to be used instead of the default. The special value keyboard indicates that keyboard input should be used.

Note

When you use the keyboard to generate random data, you must input a large amount of data. Input requiring hundreds of lines of data is not

unusual for some algorithms. The string "stop typing" appears when enough data has been input.

-t *ttl*
Specifies the time to live (TTL) value of the KEY and SIG records. The default is 3600 seconds.

-v *level*
Sets the debugging level.

key
Specifies the list of keys to be included in the keyset file. These keys are expressed in the form K*nnnn.aaa-iiiii*, which was generated by the dnssec_keygen utility.

Examples

The following command generates a keyset containing the DSA key for example.com generated in the dnssec_keygen example.

1.
```
$ dnssec_makekeyset -t 86400 -s 20000701120000 -e +2592000 -
_$ Kexample.com.003-26160
```

In this example, dnssec_makekeyset creates the file KEYSET-EXAMPLE_COM.DAT. This file contains the specified key and a self-generated signature.

The DNS administrator for example.com could send KEYSET-EXAMPLE_COM.DAT to the DNS administrator for .com for signing, if the .com zone is DNSSEC-aware and the administrators of the two zones have some mechanism for authenticating each other and for exchanging the keys and signatures securely.

dnssec_signkey

Signs keysets for DNSSEC.

Format

dnssec_signkey [-a] [-c *class*] [-s *start-time*] [-e *end-time*] [-h] [-p] [-r *randomfile*] [-v *level*] *keyset key...*

Description

The dnssec_signkey utility signs a keyset. The keyset, generated by the dnssec_makekeyset utility, is for a child zone. The child zone's keyset is signed with the zone keys for its parent zone. The output file is of the form SIGNEDKEY-*name*.DAT, where *name* is the zone name.

Parameters

keyset
Specifies the file containing the child's keyset.

key...
Specifies the keys used to sign the child's keyset.

Options

-a
Verifies all generated signatures.

-c *class*
Specifies the DNS class of the key sets.

-s *start-time*
Specifies the date and time when the generated SIG records become valid. This can be either an absolute or relative time. An absolute start time is indicated by a number in *YYYYMMDDHHMMSS* notation; 20000530144500 denotes 14:45:00 UTC on May 30, 2000. A relative start time is indicated by *+N*, which is *N* seconds from the current time. If no start time is specified, the current time is used.

-e *end-time*
Specifies the date and time when the generated SIG records expire. An absolute time is indicated in *YYYYMMDDHHMMSS* notation. A time relative to the start time is indicated by *+N*, which is *N* seconds from the start time. A time relative to the current time is indicated by now+*N*. If no end time is specified, 30 days from the start time is used as a default.

-h
Displays a short summary of the options and arguments to the dnssec_signkey command.

-p
Use pseudorandom data when signing the zone. This is faster, but less secure, than using real random data. This option may be useful when signing large zones or when the entropy source is limited.

-r *randomfile*
Specifies the source of randomness. The default source of randomness is keyboard input. *randomfile* specifies the name of a file containing random data to be used instead of the default. The special value keyboard indicates that keyboard input should be used.

Note

When you use the keyboard to generate random data, you must input a large amount of data. Input requiring hundreds of lines of data is not unusual for some algorithms. The string "stop typing" appears when enough data has been input.

-v *level*
Sets the debugging level.

Examples

The DNS administrator for a DNSSEC-aware .com zone would use the following command to sign the keyset file for example.com created by the dnssec_makekeyset utility with a key generated by the dnssec_keygen utility:

1. `$ dnssec_signkey keyset-example.com. Kcom.003-51944`

   In this example, the dnssec_signkey utility creates the file SIGNEDKEY-EXAMPLE_COM.DAT, which contains the example.com keys and the signatures by the .com keys.

dnssec_signzone

Signs a zone.

Format

dnssec_signzone [-a] [-c *class*] [-d *directory*] [-s *start-time*] [-e *end-time*] [-f *output-file*] [-h] [-i *interval*] [-n *nthreads*] [-o *origin*] [-p] [-r *randomfile*] [-t] [-v *level*] *zonefile* [*key*...]

DESCRIPTION

The dnssec_signzone utility signs a zone. It generates NXT and SIG records and produces a signed version of the zone. If there is a signedkey file from the zone's parent, the parent's signatures are incorporated into the generated signed zone file. The security status of delegations from the signed zone (that is, whether or not the child zones are secure) is determined by the presence or absence of a signedkey file for each child zone.

Before signing the zone, you must add the KEY record to the zone database file by using the $INCLUDE statement. For example, in the zone file example_com.db, add:

```
$INCLUDE Kexample_com.003-26160_KEY
```

Parameters

zonefile
Specifies the file containing the zone to be signed.

key
Specifies the keys used to sign the zone. If no keys are specified, the default is all zone keys that have private key files in the current directory.

Options

-a
Verifies all generated signatures.

-c *class*
Specifies the DNS class of the zone.

-d *directory*
Looks for signedkey files in the specified directory.

-s *start-time*
Specifies the date and time when the generated SIG records become valid. This can be either an absolute or relative time. An absolute start time is indicated by a number in *YYYYMMDDHHMMSS* notation. 20000530144500 denotes 14:45:00 UTC on May 30, 2000. A relative start time is indicated by *+N*, which is *N* seconds from the current time. If no start time is specified, the current time is used.

-e *end-time*
Specifies the date and time when the generated SIG records expire. An absolute time is indicated in *YYYYMMDDHHMMSS* notation. A time relative to the start time is indicated by *+N*, which is *N* seconds from the start time. A time relative

to the current time is indicated by now+*N*. If no end time is specified, 30 days from the start time is used as a default.

-f *output-file*
Specifies the name of the output file containing the signed zone. The default is to append _SIGNED to the input file name.

-h
Displays a short summary of the options and arguments to the dnssec_signzone command.

-i *interval*
When a previously signed zone is passed as input, records may be signed again. The interval option specifies the cycle interval as an offset from the current time (in seconds). If a SIG record expires after the cycle interval, it is retained. Otherwise, it is considered to be expiring soon, and it will be replaced.

The default cycle interval is one quarter of the difference between the signature end and start times. Therefore, if neither the end time nor the start time is specified, the dnssec_signzone utility generates signatures that are valid for 30 days, with a cycle interval of 7.5 days. Therefore, if any existing SIG records are due to expire in less than 7.5 days, they are replaced.

-n *nthreads*
Specifies the number of threads to use. By default, one thread is started for each detected CPU.

-o *origin*
Specifies the zone origin. If this option is not specified, the name of the zone file is assumed to be the origin.

-p
Uses pseudorandom data when signing the zone. This is faster, but less secure, than using real random data. This option can be useful when signing large zones or when the entropy source is limited.

-r *randomfile*
Specifies the source of randomness. The default source of randomness is keyboard input. *randomfile* specifies the name of a file containing random data to be used instead of the default. The special value keyboard indicates that keyboard input should be used.

Note

When you use the keyboard to generate random data, you must input a large amount of data. Input requiring hundreds of lines of data is not unusual for some algorithms. The string "stop typing" appears when enough data has been input.

-t
Displays statistics at completion.

-v *level*
Sets the debugging level.

Examples

The following command signs the example.com zone with the DSA key generated by the dnssec_keygen utility. The zone's keys must be in the zone. If there are signedkey files associated with this zone or any child zones, they must be in the current directory.

1. dnssec_signzone -o example.com example_com.db Kexample_com.003-26160

   In this example, dnssec_signzone creates the file EXAMPLE_COM.DB_SIGNED. This file should be referenced in a zone statement in the TCPIP$BIND.CONF file. This command displays the following:

   ```
   example_com.db_signed
   ```

rndc

Controls the operation of the BIND server.

Format

rndc [-c *config*] [-s *server*] [-p *port*] [-V] [-y *key-id*] *command*

Description

The rndc utility controls the operation of a name server. rndc communicates with the name server over a TCP connection, sending commands authenticated with digital signatures. The only supported authentication algorithm is HMAC-MD5, which uses a shared secret on each end of the connection. This provides TSIG-style authentication for the command request and the name server's response. All commands sent over the channel must be signed by a key_id known to the server.

In BIND Version 9, rndc supports all the commands of the BIND Version 8 ndc utility except start, restart, and stop. Use the BIND startup and shutdown command procedures, as described in Section C.4, to accomplish these tasks.

The rndc utility reads a configuration file to determine how to contact the name server and decide what algorithm and key it should use.

A configuration file is required, since all communication with the server is authenticated with digital signatures that rely on a shared secret, and there is no way to provide that secret other than with a configuration file. The default location for the rndc configuration file is TCPIP$ETC:RNDC.CONF, but an alternate location can be specified with the -c option. If the configuration file is not found, rndc also looks in TCPIP$ETC:RNDC.KEY. The RNDC.KEY file is generated by running rndc_confgen -a. This command provides basic functionality, but it offers less configuration flexibility than modifying the RNDC.CONF file.

Note

For the BIND server to recognize a newly generated RNDC.KEY file, you must stop and restart the BIND server.

Format of the RNDC.CONF File

The configuration file for the rndc utility is TCPIP$ETC:RNDC.CONF. The structure of this file is similar to TCPIP$BIND.CONF. Statements are enclosed in braces and are terminated with semicolons. Clauses in the statements are also terminated with semicolons. For example:

```
options {
    default-server    localhost;
    default-key       samplekey;
};

server localhost {
    key               samplekey;
};
```

```
key samplekey {
    algorithm         hmac-md5
    secret            "c3Ryb25nIGVub3VnaCBmb3IgYSBtYW";
};
```

Three statements are used in the RNDC.CONF file:

- The options statement contains three clauses:
  - The default-server clause is followed by the name or address of a name server. This host is used when the name server is not specified as an argument to rndc.
  - The default-key clause is followed by the name of a key, which is represented by a key statement and is used when the key-id is not specified on the rndc command line and when no key clause is found in a matching server statement. This key is used to authenticate the server's commands and response.
  - The default-port clause is followed by the port to connect to on the remote name server. This port is used if no -p *port* statement is supplied on the rndc command line and no port clause is included in a matching server statement.
- The server statement specifies a host name or address for a name server. The statement may include the key clause and the port clause. The key name must match the name of a key statement in the file. The port number specifies the port to connect to.
- The key statement specifies the name of a key. The statement has two clauses:
  - algorithm specifies the encryption algorithm for rndc to use (HMAC-MD5).
  - A secret clause containing the 64-bit encoding of the algorithm's encryption key. The base-64 string is enclosed in quotation marks.

    To generate the base-64 string for the secret clause, use the rndc_confgen utility. For example, enter the following command:

    ```
    $ RNDC_CONFGEN
    ```

    A complete RNDC.CONF file, including the randomly-generated key, is automatically generated. The rndc_confgen command also displays commented key and controls statements for the TCPIP$BIND.CONF file.

Commands

reload
Reloads configuration file and zones.

reload *zone* [*class* [*view*]]
Reload the given zone.

refresh *zone* [*class* [*view*]]
Schedule zone maintenance for the given zone.

reconfig
Reloads the configuration file and loads new zones, but does not reload existing zone files even if they have changed. This is faster than a full reload when there is a large number of zones because it avoids the need to examine the modification times of the zones files.

stats
Writes server statistics to the statistics file, TCPIP$BIND.STATS.

querylog
Toggles query logging. Query logging can also be enabled by explicitly directing the `queries` category to a channel in the logging section of TCPIP$BIND.CONF.

dumpdb
Dumps the server's caches to the dump file, TCPIP$BIND_DUMP.DB.

trace
Increments the servers debugging level by one.

trace *level*
Sets the server's debugging level to an explicit value.

notrace
Sets the server's debugging level to 0.

flush
Flushes the server's cache.

status
Displays the status of the server.

flush-updates
Saves any pending dynamic updates being stored in the zone journal (_JNL) files to the master zone file. (This command is available on OpenVMS systems only.)

Options

-c *config-file*
Uses *config-file* as the configuration file instead of the default, TCPIP$ETC:RNDC.CONF.

-s *server*
Specifies the name or address of the server which matches a server statement in the configuration file for `rndc`. If no server is supplied on the command line, the host named by the `default-server` clause in the `option` statement of the configuration file is used.

-p *port*
Send commands to the specified TCP port instead of the default control channel port, 953.

-V
Enables verbose logging.

-y *keyid*

Use the specified *keyid* from the configuration file specified with the `-c` option. If the configuration file was not specified on the command line, the `rndc` configuration file, RNDC.CONF, is used. The *keyid* must be known by the BIND server with the same algorithm and secret string in order for control message validation to succeed.

If no *keyid* is specified, `rndc` first looks for a `key` clause in the `server` statement of the server being used, or if no `server` statement is present for that host, then the `default-key` clause of the `options` statement.

Note that the configuration file contains shared secrets that are used to send authenticated control commands to name servers. Therefore, the file should not have general read or write access.

rndc_confgen

Generates the configuration files used by the rndc utility.

Format

rndc_confgen [-a] [-b *keysize*] [-c *keyfile*] [-h] [-k *keyname*] [-p *port*] [-r *randomfile*] [-s *address*]

Description

The rndc_confgen utility generates configuration files for the rndc utility. It can be used as a convenient alternative to writing the RNDC.CONF file and the corresponding controls and key statements in TCPIP$BIND.CONF by hand. The utility can be run with the -a option to set up an RNDC.KEY file, thereby avoiding the need for an RNDC.CONF file and a controls statement.

Options

-a
Configures rndc automatically. This option creates the file RNDC.KEY in TCPIP$ETC that is read by both rndc and the BIND server on startup. The RNDC.KEY file defines a default command channel and authentication key, allowing rndc to communicate with the BIND server with no further configuration.

Using the -a option allows BIND Version 9 and rndc to be used as drop-in replacements for BIND Version 8 and ndc, with no changes to the existing TCPIP$BIND.CONF file.

Note

For the BIND server to recognize a newly generated RNDC.KEY file, you must stop and restart the BIND server.

-b *keysize*
Specifies the size of the authentication key in bits. Must be between 1 and 512 bits; the default is 128.

-c *keyfile*
Used with the -a option to specify an alternate location for RNDC.KEY.

-h
Prints a short summary of the options and arguments to rndc_confgen.

-k *keyname*
Specifies the key name of the rndc authentication key. This must be a valid domain name. The default is rndc-key.

-p *port*
Specifies the command channel port where the BIND server listens for connections from rndc. The default is 953.

-r *randomfile*
Specifies a source of random data for generating the authorization. The default source of randomness is keyboard input. *randomfile* specifies the name of a file containing random data to be used instead of the default. The special value `keyboard` indicates that keyboard input should be used.

-s *address*
Specifies the IP address where the BIND server listens for command channel connections from `rndc`. The default is the loopback address 127.0.0.1.

C.11 Solving Bind Server Problems

To solve BIND server problems, see the following sections:

- Section C.11.1, BIND Server Diagnostic Tools
- Section C.11.2, Using NSLOOKUP to Query a Name Server
- Section C.11.3, Solving Specific Name Server Problems

C.11.1 BIND Server Diagnostic Tools

The TCP/IP Services product provides the following utilities for diagnosing problems with the BIND server:

- The `dig` utility
- The `host` utility
- The `nslookup` utility

The following sections describe these utilities.

Note

The `nslookup` utility is no longer recommended. Use the `dig` utility instead.

dig

Gathers information from the Domain Name System servers.

Format

dig [*@server*] [*-option*] [*name*] [*type*] [*class*] [*queryopt...*]

Description

dig is a flexible tool for interrogating DNS name servers. It performs DNS lookups and displays the answers that are returned from the name servers that were queried. Most DNS administrators use dig to troubleshoot DNS problems because of its flexibility, ease of use and clarity of output. Other lookup tools tend to have less functionality than dig.

Although dig normally is used with command-line arguments, it also has a batch mode of operation for reading lookup requests from a file. A brief summary of its command-line arguments and options is printed when the -h option is given. Unlike earlier versions of BIND, the BIND Version 9 implementation of dig allows multiple lookups to be issued from the command line.

Unless it is told to query a specific name server, dig tries each of the servers listed in your resolver configuration. When no command line arguments or options are given, dig performs an NS query for "." (the root).

dig has two modes: simple interactive mode, for a single query, and batch mode, which executes a query for each in a list of several query lines. All query options are accessible from the command line.

To get online help for the dig utility, enter the -h option on the command line. For example:

```
$ dig -h
```

Parameters

@server
Specifies the name or IP address of the name server to query. This can be either an IPv4 address in dotted-decimal notation or an IPv6 address in colon-delimited notation. When the supplied server argument is a host name, dig resolves that name before querying that name server. If no server argument is provided, dig consults your resolver configuration and queries the name servers listed there. The reply from the name server that responds is displayed.

name
Specifies the name of the resource record to look up.

type
Indicates the type of query required (ANY, A, MX, SIG, and so forth). If the *type* parameter is not supplied, dig performs a lookup for an A record.

class
Specifies the DNS query class. The default is class IN (Internet).

Options

-b *address*
Sets the source IP address of the query to address. This must be a valid address on one of the host's network interfaces.

-c *class*
Specifies the query class. *class* is any valid class, such as `HS` for `hesiod` records or `CH` for `CHAOSnet` records. The default query class is `IN` (Internet).

-f *filename*
Makes `dig` operate in batch mode by reading a list of lookup requests to process from the specified file. The file contains a number of queries, one per line. Each entry in the file should be organized in the same way that `dig` queries are presented using the command-line interface.

-k *filename*
Allows you to sign the DNS queries sent by `dig` and their responses using transaction signatures (TSIG). Specify a TSIG key file for *filename*.

-p *port*
Allows you to specify a nonstandard port number. *port* is the port number that `dig` uses to send its queries instead of the standard DNS port number 53. You can use this option to test a name server that has been configured to listen for queries on a nonstandard port number.

-t *type*
Sets the query type to *type*, which can be any valid query type supported in BIND Version 9. The default query type is A, unless the `-x` option is supplied to indicate a reverse lookup. A zone transfer can be requested by specifying a type of AXFR. When an incremental zone transfer (IXFR) is required, *type* is set to `ixfr=`*N*. The incremental zone transfer contains the changes made to the zone since the serial number in the zone's SOA record was *N*.

-x *addr*
Specifies reverse lookups (mapping addresses to names). *addr* is either an IPv4 address in dotted-decimal notation or a colon-delimited IPv6 address. This option eliminates the need to provide the name, class, and type arguments. `dig` automatically performs a lookup for a name like `11.12.13.10.in-addr.arpa` and sets the query type and class to PTR and IN, respectively. By default, IPv6 addresses are looked up using the IP6.ARPA domain and binary labels as defined in RFC 2874. To use the older RFC 1886 method using the IP6.INT domain and nibble labels, specify the `-n` (nibble) option.

-y *name:key*
Allows you to specify the TSIG key itself on the command line. *name* is the name of the TSIG key and *key* is the actual key. The key is a base-64 encoded string, typically generated by `dnssec_keygen`. When using TSIG authentication with `dig`, the name server that is queried needs to know the key and algorithm that is being used. In BIND, this is done by providing appropriate key and server statements in TCPIP$BIND.CONF.

Query Options

Each query option is identified by a keyword preceded by a plus sign (+). Some keywords set or reset an option. These can be preceded by the string no to negate the meaning of that keyword. Other keywords (like that which sets the timeout interval) assign values to options. These types of keywords have the form *+keyword=value*.

The query options are:

+[no]tcp
Specifies whether to use TCP when querying name servers. The default behavior is to use UDP unless an AXFR or IXFR query is requested, in which case a TCP connection is used.

+[no]vc
Specifies whether to use TCP when querying name servers. This alternate syntax to [no]tcp is provided for backward compatibility. (vc stands for virtual circuit.)

+[no]ignore
Ignores truncation in UDP responses instead of retrying with TCP. By default, TCP retries are performed.

+domain=*name*
Sets the search list to contain the single domain *name*, as if specified in a domain directive in your resolver configuration. Enables search list processing as if the search option were specified.

+[no]search
Specifies whether to use the search list defined by the path directive in your resolver configuration. By default, the search list is not used.

+[no]defname
This deprecated option is treated as a synonym for [no]search.

+[no]aaonly
This option does nothing. It is provided for compatibility with old versions of dig, in which it set an unimplemented resolver flag.

+[no]adflag
Specifies whether to set the AD (authentic data) bit in the query. The AD bit currently has a standard meaning only in responses, not in queries, but the ability to set the bit in the query is provided for completeness.

+[no]cdflag
Specifies whether to set the CD (checking disabled) bit in the query. This requests the server to not perform DNSSEC validation of responses.

+[no]recursive
Toggles the setting of the RD (recursion desired) bit in the query. This bit is set by default, which means dig normally sends recursive queries. Recursion is automatically disabled when the nssearch or trace query options are used.

+[no]nssearch
Attempts to find the authoritative name servers for the zone containing the name being looked up. Displays the SOA record that each name server has for the zone.

+[no]trace
Toggles tracing of the delegation path from the root name servers for the name being looked up. Tracing is disabled by default. When tracing is enabled, `dig` makes iterative queries to resolve the name being looked up, following referrals from the root servers and showing the answer from each server that was used to resolve the lookup.

+[no]cmd
Toggles the printing of the initial comment in the output identifying the version of `dig` and the query options that have been applied. This comment is printed by default.

+[no]short
Provides a terse answer. The default is to print the answer in verbose form.

+[no]identify
Specifies whether to show the IP address and port number that supplied the answer when the `+short` option is enabled. If terse answers are requested, the default is not to show the source address and port number of the server that provided the answer.

+[no]comments
Toggles the display of comment lines in the output. The default is to print comments.

+[no]stats
Toggles the printing of statistics, such as when the query was made, the size of the reply, and so on. The default behavior is to print the query statistics.

+[no]qr
Specifies whether to print the query as it is sent. By default, the query is not printed.

+[no]question
Specifies whether to print the question section of a query when an answer is returned. The default is to print the question section as a comment.

+[no]answer
Specifies whether to display the answer section of a reply. The default is to display the answer section.

+[no]authority
Specifies whether to display the authority section of a reply. The default is to display the authority section.

+[no]additional
Specifies whether to display the additional section of a reply. The default is to display the additional section.

+[no]all
Specifies whether to set or clear all display flags.

+time=*T*
Sets the timeout for a query to *T* seconds. The default timeout is 5 seconds. An attempt to set *T* to less than 1 results in a query timeout of 1 second.

+tries=*A*
Sets the number of times to retry UDP queries to the server to *A* instead of to the default of 3. If *A* is less than or equal to zero, the number of retries is silently rounded up to 1.

+ndots=*D*
Set the number of dots that have to appear in name to *D* for it to be considered absolute. The default value is 1.

Names with fewer dots are interpreted as relative names and are searched for in the domains listed in the search or domain directive in your resolver configuration.

+bufsize=*B*
Set the UDP message buffer size advertised using EDNS0 to *B* bytes. The maximum and minimum sizes of this buffer are 65535 and 0, respectively. Values outside this range are rounded up or down appropriately.

+[no]multiline
Prints records like the SOA records in a verbose multiline format with human-readable comments. The default is to print each record on a single line, to facilitate machine parsing of the output.

+[no]fail
Does not continue querying the next server if a SERVFAIL error occurs.

+[no]besteffort
Attempts to parse even illegal messages.

+[no]dnssec
Requests DNSSEC records.

Multiple Queries

The BIND Version 9 implementation of `dig` supports the specification of multiple queries on the command line (in addition to supporting the `-f` batch file option). Each of those queries can be supplied with its own set of flags, options, and query options.

Each query argument represent an individual query in the command-line syntax. Each individual query consists of any of the standard options and flags, the name to be looked up, an optional query type and class, and any query options that are needed.

A global set of query options, which should be applied to all queries, can also be supplied. These global query options must precede the first tuple of name, class, type, options, flags, and query options supplied on the command line. Any global query options (except for the `+[no]cmd` option) can be overridden by a query-specific set of query options.

Examples

The following example shows how to use `dig` from the command line to make three lookups:

1. An ANY query for `www.isc.org`
2. A reverse lookup of 127.0.0.1

3. A query for the NS records of isc.org.

1. dig +qr www.isc.org any -x 127.0.0.1 isc.org ns +noqr

```
; <<>> DiG 9.2.0 <<>> +qr www.isc.org any -x 127.0.0.1 isc.org ns +noqr
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 38437
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 5, ADDITIONAL: 5

;; QUESTION SECTION:
;www.isc.org.                   IN      ANY

;; ANSWER SECTION:
www.isc.org.            3421    IN      CNAME   isc.org.

;; AUTHORITY SECTION:
isc.org.                3421    IN      NS      gns1.nominum.com.
isc.org.                3421    IN      NS      gns2.nominum.com.
isc.org.                3421    IN      NS      ns-ext.vix.com.
isc.org.                3421    IN      NS      ns-int.vix.com.
isc.org.                3421    IN      NS      ns1.gnac.com.

;; ADDITIONAL SECTION:
ns1.gnac.com.           17389   IN      A       209.182.195.77
gns1.nominum.com.       92      IN      A       198.133.199.1
gns2.nominum.com.       68661   IN      A       198.133.199.2
ns-ext.vix.com.         2601    IN      A       204.152.184.64
ns-int.vix.com.         828     IN      A       204.152.184.65

;; Query time: 134 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Tue Nov  6 13:09:16 2001
;; MSG SIZE  rcvd: 241

;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 16441
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 1

;; QUESTION SECTION:
;1.0.0.127.in-addr.arpa.                IN      PTR

;; ANSWER SECTION:
1.0.0.127.in-addr.arpa. 86400   IN      PTR     localhost.

;; AUTHORITY SECTION:
0.0.127.in-addr.arpa.   86400   IN      NS      localhost.

;; ADDITIONAL SECTION:
localhost.              86400   IN      A       127.0.0.1

;; Query time: 224 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Tue Nov  6 13:09:16 2001
;; MSG SIZE  rcvd: 93

;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 9922
;; flags: qr rd ra; QUERY: 1, ANSWER: 5, AUTHORITY: 0, ADDITIONAL: 5

;; QUESTION SECTION:
;isc.org.                       IN      NS

;; ANSWER SECTION:
isc.org.                3421    IN      NS      ns-ext.vix.com.
isc.org.                3421    IN      NS      ns-int.vix.com.
isc.org.                3421    IN      NS      ns1.gnac.com.
isc.org.                3421    IN      NS      gns1.nominum.com.
isc.org.                3421    IN      NS      gns2.nominum.com.
```

```
;; ADDITIONAL SECTION:
ns1.gnac.com.           17389   IN      A       209.182.195.77
gns1.nominum.com.       92      IN      A       198.133.199.1
gns2.nominum.com.       68661   IN      A       198.133.199.2
ns-ext.vix.com.         2601    IN      A       204.152.184.64
ns-int.vix.com.         828     IN      A       204.152.184.65

;; Query time: 198 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Tue Nov  6 13:09:17 2001
;; MSG SIZE  rcvd: 223
```

A global query option of +qr is applied so that dig shows the initial query it made for each lookup. The final query has a local query option of +noqr, which means that dig will not print the initial query when it looks up the NS records for isc.org.

The final portion of the output displays the following information:

- Amount of time the query took
- Server and port to which the query was sent, in the form *server#port*
- Date and time of the query
- Message size

host

The host utility allows you to look up Internet host names. By default, the host utility converts between host names and Internet addresses, but its functionality can be extended with the use of options.

Format

host [-aCdlrTvw] [-c *class*] [-n] [-N *ndots*] [-R *number*] [-t *type*] [-W *wait*] *name* [*server*]

Description

The host utility is used to convert names to IP addresses and vice versa. When no arguments or options are given, the host utility prints a short summary of its command line arguments and options.

Parameters

name
Specifies the domain name that is to be looked up. It can also be a dotted-decimal IPv4 address or a colon-delimited IPv6 address, in which cases the host performs a reverse lookup for that address by default.

[*server*]
Specifies the name or IP address of the name server that the host utility should query instead of the server or servers listed in your resolver configuration.

Options

-a
Equivalent to setting the -v option and asking the host utility to make a query of type ANY.

-C
Displays the SOA records for zone *name* from all the listed authoritative name servers for that zone. The list of name servers is defined by the NS records that are found for the zone. The -C option must be enclosed in quotation marks. For example:

```
$ host -"C" name
```

-c *class*
Makes a DNS query of class *class*. This can be used to look up hesiod or CHAOSnet class resource records. The default class is IN (Internet).

-d
Specifies verbose output.

-l
Selects list mode. This makes the host utility perform a zone transfer for zone *name*. The argument is provided for compatibility with older implemementations. This option is equivalent to making a query of type AXFR.

-n
Specifies that reverse lookups of IPv6 addresses should use the IP6.INT domain and nibble labels, as defined in RFC 1886. The default is to use IP6.ARPA and binary labels, as defined in RFC 2874.

-N *number*
Sets the number of dots that have to be in the zone name for it to be considered absolute. The default value is 1. Names with fewer dots are interpreted as relative names and are searched for in the domains listed in the search path defined in the resolver configuration.

-R *number*
Changes the number of UDP retries for a lookup. The value for *number* indicates how many times the host utility repeats a query that does not get answered. The default number of retries is 1. If *number* is negative or zero, the number of retries defaults to 1.

-r
Makes nonrecursive queries. Setting this option clears the RD (recursion desired) bit in the query that the host utility makes. This should mean that the name server receiving the query does not attempt to resolve *name*. The -r option enables host to mimic the behavior of a name server by making nonrecursive queries and expecting to receive answers to those queries that are usually referrals to other name servers.

-T
Uses a TCP connection when querying the name server. By default, the host utility uses UDP when making queries.

TCP is automatically selected for queries that require it, such as zone transfer (AXFR) requests.

-t *type*
Selects the query type. *type* can be any recognized query type, such as CNAME, NS, SOA, SIG, KEY, or AXFR. When no query type is specified, the host utility automatically selects an appropriate query type. By default, the host utility looks for A records, but if the -C option is specified, queries are made for SOA records. If *name* is a dotted-decimal IPv4 address or a colon-delimited IPv6 address, the host utility queries for PTR records.

-v
Generates verbose output.

-W *wait*
Makes the host utility wait for the number of seconds specified by *wait* before making the query. If *wait* is less than 1, the wait interval is set to 1 second.

-w
Waits forever for a reply. The time to wait for a response is set to the number of seconds given by the hardware's maximum value for an integer quantity.

C.11.2 Using NSLOOKUP to Query a Name Server

The nslookup utility is a debugging tool provided with BIND that allows anyone to directly query a name server and retrieve information. Use NSLOOKUP to determine whether your local name server is running correctly or to retrieve information from remote servers.

nslookup makes direct queries to name servers around the world to obtain DNS information, which includes the following:

- Host names and addresses on the local domain
- Host names and addresses on remote domains
- Host names that serve as Mail Exchange (MX) records
- Name servers for a specific zone

Note

The nslookup utility is deprecated. Compaq recommends that you use the dig utility instead.

For online information about using the nslookup utility, enter the following command:

```
$ HELP TCPIP_SERVICES NSLOOKUP
```

C.11.3 Solving Specific Name Server Problems

The following sections describe some problems commonly encountered with BIND and how to solve them.

C.11.3.1 Server Not Responding

A missing client name in the BIND server's database files results in lack of service to that client. If records that point to the name servers (NS records) in a domain are missing from your server's database files, you might see the following messages:

```
%TCPIP-W-BIND_NOSERVNAM, Server with address 199.85.8.8 is not responding
%TCPIP-E-BIND_NOSERVERS, Default servers are not available
%TCPIP-W-NORECORD, Information not found
-TCPIP-E-BIND_NOSERVERS, Default servers are not available
```

When the CONVERT/ULTRIX BIND /DOMAIN command creates the .DB files from the hosts database, it cannot detect the existence or the names of name servers in a domain. Therefore, it does not add NS records for the name servers to the .DB files.

To solve the problem, follow these steps:

1. Stop the BIND server.
2. Manually add NS records for the missing names.
3. Update the start-of-authority (SOA) records by incrementing the serial number.
4. Restart the BIND server.

D

Advanced IPv6 Programming Socket Interface

This appendix describes:

- The data structures for sending and receiving ancillary data (Section D.1)
- How to use IPv6 raw socket (Section D.2)
- The socket calls used to build and examine IPv6 options headers (Section D.3)
- The socket calls used to build and examine IPv6 routing headers (Section D.4)

D.1 Socket-Related Data Structures for Sending and Receiving Ancillary Data

The following data structures enable applications to send and receive ancillary data using the sendmsg and recvmsg system calls:

- struct msghdr

  This data structure, which is defined in the *sys/socket.h* header file, also allows AF_INET sockets and raw AF_INET6 sockets to receive certain data.

  For IPv4, see the *Compaq TCP/IP Services for OpenVMS Sockets API and System Services Programming* manual for the descriptions of the IP_RECVDSTADDR and IP_RECVOPTS options.

  For IPv6, this section describes the IPV6_RECVHOPOPTS, IPV6_RECVDSTOPTS, and IPV6_RECVRTHDR options.

  The msghdr data structure consists of the following components:

```
struct msghdr {
        void *msg_name;  /* optional address */
        size_t msg_namelen; /* size of address */
        struct iovec *msg_iov; /* scatter/gather array */
        int msg_iovlen;  /* # elements in msg_iov */
        void *msg_control; /* ancillary data, see below */
        size_t msg_controllen; /* ancillary data buffer len */
        int msg_flags;  /* flags on received message */
};
```

- struct cmsghdr

  Describes ancillary data objects transferred by the sendmsg and recvmsg system calls.

  The msg_control member of the msghdr data structure points to the ancillary data that are contained in a cmsghdr structure. Typically, only one data object is passed in a cmsghdr structure. However, the IPv6 advanced sockets API enables the sendmsg and recvmsg system calls to pass multiple objects. See Section D.2 for information about using raw IPv6 sockets.

  The data structure is defined in the sys/socket.h header file.

The cmsghdr data structure consists of the following components:

```
struct cmsghdr {
        socklen_t cmsg_len; /* #bytes, including this header */
        int cmsg_level;  /* originating protocol */
        int cmsg_type;  /* protocol-specific type */
        /* followed by unsigned char cmsg_data[]; */
};
```

D.2 Using IPv6 Raw Sockets

Raw sockets are used in both IPv4 and IPv6 to bypass the TCP and UDP transport layers.

Table D–1 describes the principal differences between IPv4 and IPv6 raw sockets.

Table D–1 Differences Between IPv4 and IPv6 Raw Sockets

	IPv4	IPv6
Use	Access ICMPv4, IGMPv4, and to read and write IPv4 datagrams that contain a protocol field the kernel does not recognize.	Access ICMPv6,and to read and write IPv6datagrams that contain a Next Header field the kernel does not recognize.
Byte order	Not specified.	Network byte order for all data sent and received.
Send and receive complete packets	Yes	No. Uses ancillary data objects to transfer extension headers and hop limit information.

For output, applications can modify all fields, except for the flow label field, by using ancillary data or socket options, or both.

For input, applications can access all fields, except for the flow label, version number, and Next Header fields, and all extension headers by using ancillary data.

For IPv6 raw sockets other than ICMPv6 raw sockets, the application must set the IPV6_CHECKSUM socket option. For example:

```
int offset = 2;
setsockopt (fd, IPPROTO_IPV6, IPV6_CHECKSUM, &offset, sizeof(offset));
```

This enables the kernel to compute and store a checksum for output and to verify the checksum on input. This relieves the application from having to perform source address selection on all outgoing packets. This socket option is disabled by default. You can disable this option by setting the offset variable to -1.

Using IPv6 raw sockets, an application can access the following information:

- ICMPv6 messages
- IPv6 header

- Routing header
- IPv6 options headers: hop-by-hop options header and destination options header

The following sections describe how to access this information.

D.2.1 Accessing ICMPv6 Messages

An ICMPv6 raw socket is a socket that is created by calling the socket function with the PF_INET6, SOCK_RAW, and IPPROTO_ICMPV6 arguments.

The kernel calculates and inserts the ICMPv6 checksum for all outbound ICMPv6 packets and verifies the checksum for all received packets. If the received checksum is incorrect, the packet is discarded.

Because ICMPv6 is a superset of ICMPv4,an ICMPv6 raw socket can receive many more messages than an ICMPv4 raw socket. By default, when you create an ICMPv6 raw socket, it passes all ICMPv6 message types to an application. An application, however, does not need access to all messages. An application can specify the ICMPv6 message types it wants passed by creating an ICMPv6 filter.

The ICMPv6 filter has a datatype of struct icmp6_filter. Use getsockopt to retrieve the current filter and setsockopt to store the filter. For example, to enable filtering of ICMPv6 messages, use the ICMP6_FILTER option, as follows:

```
struct icmp6_filter myfilter;

setsockopt (fd, IPPROTO_ICMPV6, IPV6_FILTER, &(myfilter), (sizeof)(myfilter));
```

The value of *myfilter* is an ICMPv6 message type between 0 and 255.

Table D–2 describes the ICMPv6 filter macros.

Table D–2 ICMPv6 Filtering Macros

Macro	Description
ICMP6_FILTER_SETPASSALL	Passes all ICMPv6 messages to an application.
ICMP6_FILTER_SETBLOCKALL	Blocks all ICMPv6 messages from being passed to an application.
ICMP6_FILTER_SETPASS	Passes ICMPv6 messages of a given type to an application.
ICMP6_FILTER_SETBLOCK	Blocks ICMPv6 messages of a given type from being passed to an application.
ICMP6_FILTER_WILLPASS	Returns true, if specified message type is passed to application.
ICMP6_FILTER_WILLBLOCK	True, if the specified message type is blocked from being passed to an application.

To clear an installed filter, call setsockopt for the ICMP_FILTER option with a zero-length filter.

The kernel does not perform any validity checks on message type, message content, or packet structure. The application is responsible for checking them.

D.2.2 Accessing the IPv6 Header

When using IPv6 raw sockets, applications must be able to receive the IPv6 header content. To receive this optional information, use the setsockopt system call with the appropriate socket option.

Table D–3 describes the socket options for receiving optional information.

Table D–3 Optional Information and Socket Options

Optional Information	Socket Option	cmsg_type
Source and destination IPv6 address, and sending and receiving interface	IPV6_RECVPKTINFO	IPV6_PKTINFO
Hop limit	IPV6_RECVHOPLIMIT	IPV6_HOPLIMIT
Routing header	IPV6_RECVRTHDR	IPV6_RTHDR
Hop-by-Hop options	IPV6_RECVHOPOPTS	IPV6_HOPOPTS
Destination options	IPV6_RECVDSTOPTS	IPV6_DSTOPTS

The recvmsg system call returns the received data as one or more ancillary data objects in a cmsghdr data structure.

To determine the value of a socket option, use the getsockopt system call with the corresponding option. If the IPV6_RECVPKTINFO option is not set, the function returns an in6_pktinfo data structure with ipi6_addr set to in6addr_any and ipi6_addr set to zero. For other options, the function returns an option_len value of zero if there is no option value.

An application can receive the following IPv6 header information as ancillary data from incoming packets:

- Destination IPv6 address
- Interface index
- Hop limit

The IPv6 address and interface index are contained in a in6_pktinfo data structure that is received as ancillary data with the recvmsg system call. the in6_pktinfo data structure is defined in netinet/in.h. The tasks associated with the IPv6 header are:

- Receiving an IPv6 address

  If the IPV6_RECVPKTINFO option is enabled, the recvmsg system call returns a in6_pktinfo data structure as ancillary data. The ipi6_addr member contains the destination IPv6 address from the received packet. For TCP sockets, the destination address is the local address of the connection.

- Receiving an interface

  If the IPV6_RECVPKTINFO option is enabled, the recvmsg system call returns a in6_pktinfo data structure as ancillary data. The ipi6_ifindex member contains the interface index of the interface that received the packet.

- Receiving a hop limit

If the IPV6_RECVHOPLIMIT option is enabled, the recvmsg system call returns a cmsghdr data structure as ancillary data. The cmsg_type member is IPV6_HOPLIMIT and the cmsg_data[] member contains the first byte of the integer hop limit.

D.2.3 Accessing the IPv6 Routing Header

The advanced sockets API enables you to access the IPv6 routing header. The routing header is an IPv6 extension header that enables an application to perform source routing. RFC 2460 defines the type 0 routing header, which supports up to 127 intermediate nodes, or 128 hops.

Table D–4 describes the sockets calls that an application uses to build and examine routing headers.

Table D–4 Socket Calls for Routing Header Name Description

Name	Description
inet6_rth_space	Returns the number of bytes required for a routing header.
inet6_rth_init	Initializes buffer data for a routing header.
inet6_rth_add	Adds one address to a routing header.
inet6_rth_reverse	Reverses the order of fields in a routing header.
inet6_rth_segments	Returns the number of segments, or addresses, in a routing header.
inet6_rth_getaddr	Fetches one address from a routing header.

The tasks associated with the routing header are:

- Receiving a routing header

  To receive a routing header, an application calls setsockopt with the IPV6_RECVRTHDR option enabled. See Section D.4 for more information.

  For each received routing header, the kernel passes one ancillary data object in a cmsghdr structure with the cmsg_type member set to IPV6_RTHDR. An application processes a routing header by calling inet6_rth_reverse, inet6_rth_segments, and inet6_rth_getaddr.

- Sending a routing header

  To send a routing header, an application specifies the header either as ancillary data in a call to sendmsg or by calling setsockopt. An application can remove a sticky routing header by calling setsockopt for the IPV6_RTHDR option and specifying a option length of zero.

  When using ancillary data,the application sets cmsg_level member to IPPROTO_IPV6 and the cmsg_type member to IPV6_RTHDR. Use the inet6_rth_space, inet6_rth_init, and inet6_rth_add calls to build the routing header. See Section D.4 for more information.

  When an application specifies a routing header, the destination address specified in a call to the connect sendto or sendmsg function is the final destination of the datagram. Therefore, the routing header contains the addresses of all intermediate nodes.

  Because of the order of extension headers specified in RFC 2460, an application can specify only one outgoing routing header.

D.2.4 Accessing the IPv6 Options Headers

The advanced sockets API enables applications to access the following IPv6 options headers:

- Hop-by-hop header

  A single hop-by-hop options header can contain a variable number of hop-by-hop options. Each option is encoded with a type, length, and value (TLV). The application uses sticky options or ancillary data to communicate this information with the kernel.

- Destination header

  One or more destination options headers can contain a variable number of destination options. A destination options header appearing before a routing header is processed by the first and subsequent destinations specified in the routing header. A destination option appearing after the routing header is processed only by the final destination. Each option is encoded with a type, length, and value (TLV). The application uses sticky options or ancillary data to communicate this information with the kernel.

See RFC 2460 for additional information about the alignment requirements of the headers and ordering of the extensions headers.

Table D–5 lists the sockets calls that an application uses to build and examine hop-by-hop and destination headers.

Table D–5 Socket Calls for Options Headers

Socket Call	Description
inet6_opt_init	Initializes buffer data for options.
inet6_opt_append	Adds an option to the options header.
inet6_opt_finish	Finishes adding options to the options header.
inet6_opt_set_val	Adds one component of the option content to the options header.
inet6_opt_next	Extracts the next option from the options header.
inet6_opt_find	Extracts an option of a specified type from the options header.
inet6_opt_get_val	Retrieves one component of the option content from the options header.

The tasks associate with options headers are:

- Receiving hop-by-hop options

  To receive a hop-by-hop options header, an application calls setsockopt with the IPV6_RECVHOPOPTS option enabled.

  When using ancillary data, the kernel passes a hop-by-hop options header to the application and sets the cmsg_level member to IPPROTO_IPV6 and the cmsg_type member to IPV6_HOPOPTS.

  An application retrieves these options by calling inet6_opt_next, inet6_opt_find, and inet6_opt_get_val. See Section D.3 for more information

- Sending hop-by-hop options

To send a hop-by-hop options header, an application specifies the header either as ancillary data in a call to sendmsg or by calling setsockopt An application can remove a sticky hop-by-hop options header by calling setsockopt for the IPV6_HOPOPTS option and specifying a option length of zero (0).

When using ancillary data, all hop-by-hop options are specified by a single ancillary data object. The application sets cmsg_level member to IPPROTO_IPV6 and the cmsg_type member to IPV6_HOPOPTS. Use the inet6_opt_init, inet6_opt_append, inet6_opt_finish, and inet6_opt_set_val calls to build the option header. See Section D.3 for more information.

- Receiving destination options

  To receive a destination options header, an application calls setsockopt with the IPV6_RECVDSTOPTS option enabled. The kernel passes each destination option to the application as one ancillary data object and sets the cmsg_level member to IPPROTO_IPV6 and the cmsg_type member to IPV6_DSTOPTS.

  An application processes these options by calling inet6_opt_next, inet6_opt_find, and inet6_opt_get_val. See Section D.3 for more information.

- Sending destination options

  To send a destination options header, an application specifies the header either as ancillary data in a call to sendmsg or by calling setsockopt.

  An application can remove a sticky hop-by-hop options header by calling setsockopt for either the IPV6_RTHDRDSTOPTS or the IPV6_DSTOPTS option and specifying a option length of zero (0).

  In accordance with RFC 2460, the API assumes that the extension headers are in order. Only one set of destination options can precede a routing header and only one set of destination options can follow a routing header.

  Each set can contain one or more options, but each set is considered a single extension header.

  When using ancillary data, the application passes a destination options header to the kernel in one of the following ways:

  – For destination options that precede a routing header, the application sets the cmsg_level member to IPPROTO_IPV6 and the cmsg_type member to IPV6_RTHDRDSTOPTS. Any setsockopt or ancillary data is ignored unless the application explicitly specifies its own routing header.

  – For destination options that follow a routing header or when no routing header is specified, the application sets the cmsg_level member to IPPROTO_IPV6 and the cmsg_type member to IPV6_DSTOPTS.

  An application builds these options by calling inet6_opt_init, inet6_opt_append, inet6_opt_finish, and inet6_opt_set_val. See Section D.3 for more information.

D.3 Socket Calls to Build and Examine IPv6 Options Headers

The socket calls that are used to build and examine IPv6 option headers are:

- inet6_opt_append
- inet6_opt_find
- inet6_opt_finish
- inet6_opt_get_val
- inet6_opt_init
- inet6_opt_next
- inet6_opt_set_val

D.3.1 The inet6_opt_append Socket Call

Returns the length of an IPv6 extension header with a new option and appends the option.

```
#include <netinet/ip6.h>

int inet6_opt_apppend(void *extbuf, size_t extlen, int prevlen,
          uint8_t type, size_t len, uint_t align, void **databufp );
```

D.3.1.1 Parameters

- extbuf

  Points to a buffer that contains an extension header. This is either a valid pointer or a NULL pointer.

- extlen

  Specifies the length of the extension header to initialize. Valid values are 0 if _extbuf equals 0, a value returned by inet6_opt_finish(), or any number that is a multiple of 8.

- prevlen

  Specifies the length of the existing extension header. Obtain this value from a prior call to inet6_opt_init() or inet6_opt_append().

- type

  Specifies the type of option. Specify a value from 2 to 255, inclusive, excluding 194.

- len

  Specifies the length of the option data, excluding the option type and option length fields. Specify a value from 0 to 255, inclusive.

- align

  Specifies the alignment of the option. Specify one of the following values: 1, 2, 4, or 8.

- databufp

  Points to a buffer that contains the option data.

D.3.1.2 Description

The inet6_opt_append() function, when called with extbuf as a NULL pointer and extlen as 0, returns the updated number of bytes in an extension header.

If you specify extbuf as a valid pointer and valid extlen and align parameters, the function returns the same information as in the previous case, but also inserts the pad option, initializes the type and length fields, and returns a pointer to the location for the option content.

After you call inet6_opt_append(), you can then use the data buffer directly or call inet6_optt_set_val() to specify the option contents.

D.3.1.3 Return Values

Upon successful completion, the inet6_opt_append() function returns the updated number of bytes in an extension header.

Upon failure, it returns a -1.

D.3.2 The inet6_opt_find Call

Finds a specific option in an extension header.

```
#include <netinet/ip6.h>

int inet6_opt_find(
        void *extbuf, size_t extlen, int prevlen, uint8_t type,
        size_t *lenp, void **databufp );
```

D.3.2.1 Parameters

- extbuf

  Points to a buffer that contains an extension header.

- extlen

  Specifies the length, in bytes, of the extension header.

- prevlen

  Specifies the location in the extension header of an option. Valid values are either 0 (zero) for the first option or the length returned from a previous call to either inet6_opt_next() or inet6_opt_find().

- type

  Specifies the type of option to find.

- lenp

  Points to the length of the option found.

- databufp

  Points to the option data.

D.3.2.2 Description

The inet6_opt_find() function searches a received option extension header for an option specified by type. If it finds the specified option, it returns the option length and a pointer to the option data. In addition, it returns an offset to the next option that you specify in the prevlen parameter to subsequent calls to inet6_opt_next() in order to search for additional occurrences of the same option type.

D.3.2.3 Return Values

Upon successful completion, the inet6_opt_find() function returns an offset from which you can begin the next search in the data buffer.

Upon failure, it returns a -1.

D.3.3 The inet6_opt_finish Call

Returns the total length of an IPv6 extension header, including padding, and initializes the option.

```
#include <netinet/ip6.h>

int inet6_opt_finish(
        void *extbuf, size_t extlen, int prevlen );
```

D.3.3.1 Parameters

- extbuf

  Points to a buffer that contains an extension header. This is either a valid pointer or a NULL pointer.

- extlen

  Specifies the length of the extension header to finish initializing. A valid value is any number greater than or equal to 0.

- prevlen

  Specifies the length of the existing extension header. Obtain this value from a prior call to inet6_opt_init() or inet6_opt_append().

D.3.3.2 Description

The inet6_opt_finish() function when called with extbuf as a NULL pointer and extlen as 0, returns the total number of bytes in an extension header, including final padding.

If you specify extbuf as a valid pointer and a valid extlen parameter, the function returns the same information as in the previous case, increments the buffer pointer, and verifies that the buffer is large enough to hold the header.

D.3.3.3 Return Values

Upon successful completion, the inet6_opt_finish() function returns the total number of bytes in an extension header, including padding.

Upon failure, it returns a -1.

D.3.4 The inet6_opt_get_val Call

Extracts data items from the data portion of an IPv6 option.

```
#include <netinet/ip6.h>

int inet6_opt_get_val(
        void *databuf, size_t offset, void *val, int vallen );
```

D.3.4.1 Parameters

- databuf

  Points to a buffer that contains an extension header. This is a pointer returned by a call to inet6_opt_find() or inet6_opt_next().

- offset

  Specifies the location in the data portion of the option from which to extract the data. You can access the first byte after the option type and length by specifying the offset of 0.

- val

  Points to a destination for the extracted data.

- vallen

  Specifies the length of the data, in bytes, to be extracted.

D.3.4.2 Description

The inet6_opt_get_val() function copies data items from data buffer databuf beginning at offset to the location val. In addition, it returns the offset for the next data field to assist you in extracting option content that has multiple fields.

Make sure that each field is aligned on its natural boundaries.

D.3.4.3 Return Values

Upon successful completion, the inet6_opt_get_val() unction returns the offset for the next field in the data buffer.

Upon failure, it returns a -1.

D.3.5 The inet6_opt_init Call

Returns the length of an IPv6 extension header with no options and initializes the header.

```
#include <netinet/ip6.h>

int inet6_opt_innit( void *extbuf, size_t extlen );
```

D.3.5.1 Parameters

- extbuf

  Points to a buffer that contains an extension header. This is either a valid pointer or a NULL pointer.

- extlen

  Specifies the length of the extension header to initialize. Valid values are 0 and any number that is a multiple of 8.

D.3.5.2 Description

The inet6_opt_init() unction when called with extbuf as a NULL pointer and extlen as 0, returns the number of bytes in an extension header that has no options.

If you specify extbuf as a valid pointer and extlen as a number that is a multiple of 8, the function returns the same information as in the previous case, initializes the extension header, and sets the length field.

Return Values

Upon successful completion, the inet6_opt_init() function returns the number of bytes in an extension header with no options.

Upon failure, it returns a -1.

D.3.6 The inet6_opt_next Call

Parses received option extension headers.

```
#include <netinet/ip6.h>

int inet6_opt_next(
        void *extbuf, size_t extlen, int prevlen, uint8_t *typep,
        size_t *lenp, void **databufp );
```

D.3.6.1 Parameters

- extbuf

  Points to a buffer that contains an extension header.

- extlen

  Specifies the length, in bytes, of the extension header.

- prevlen

  Specifies the location in the extension header of an option. Valid values are either 0 for the first option or the length returned from a previous call to either inet6_opt_next() or inet6_opt_find().

- typep

  Points to the type of the option found.

- lenp

  Points to the length of the option found.

- databufp

  Points to the option data.

D.3.6.2 Description

The inet6_opt_next() function parses a received option extension header and returns the next option. In addition, it returns an offset to the next option that you specify in the prevlen parameter to subsequent calls to inet6_opt_next().

This function does not return any PAD1 or PADN options.

D.3.6.3 Return Values

Upon successful completion, the inet6_opt_next() function returns the offset for the next option in the data buffer.

Upon failure, it returns a -1.

D.3.7 The inet6_opt_set_val Call

Inserts data items into the data portion of the IPv6 option.

```
#include <netinet/ip6.h>

int inet6_opt_set_val(void *databuf, size_t offset, void *val, int vallen );
```

D.3.7.1 Parameters

- `databuf`

  Points to a buffer that contains an extension header. This is a pointer returned by a call to `inet6_opt_append()`.

- `offset`

  Specifies the location in the data portion of the option into which to insert the data. You can access the first byte after the option type and length by specifying the offset of 0 (zero).

- `val`

  Points to the data to be inserted.

- `vallen`

  Specifies the length of the data, in bytes, to be inserted.

D.3.7.2 Description

The `inet6_opt_set_val()` function copies data items at the location `val` into a data buffer `databuf` beginning at `offset`. In addition, it returns the offset for the next data field to assist you in composing content that has multiple fields.

Make sure that each field is aligned on its natural boundaries.

D.3.7.3 Return Values

Upon successful completion, the `inet6_opt_set_val()` function returns the offset for the next field in the data buffer.

Upon failure, it returns a -1.

D.4 Socket Calls to Build and Examine IPv6 Routing Headers

The socket call used to build and examine IPv6 routing headers are:

- `inet6_rth_add`
- `inet6_rth_getaddr`
- `inet6_rth_init`
- `inet6_rth_reverse`
- `inet6_rth_segments`
- `inet6_rth_space`

D.4.1 The inet6_rth_add Call

Adds an IPv6 address to the routing header under construction.

```
#include <netinet/ip6.h>

int inet6_rth_add( void *bp, const struct in6_addr *addr );
```

D.4.1.1 Parameters

- `bp`

  Points to a buffer that is to contain an IPv6 routing header.

- `addr`

  Points to an IPv6 address to add to the routing header.

D.4.1.2 Description

The inet6_rth_add() function adds IPv6 address to the end of the routing header under construction. The address pointed to by addr cannot be either an IPv6 V4-mapped address or an IPv6 multicast address.

The function increments the ip60r_segleft member in the ip6_rthdr0 structure. The ip6_rthdr0 structure is defined in netinet/ip6.h.

Only routing header type 0 is supported.

D.4.1.3 Return Values

Upon successful completion, the inet6_rth_add() function returns 0 (zero).

Upon failure, it returns a -1.

D.4.2 The inet6_rth_getaddr Call

Retrieves an address for an index from an IPv6 routing header.

```
#include <netinet/ip6.h>

struct in6_addr *inet6_rth_getaddr( const void *bp,int index );
```

D.4.2.1 Parameters

- bp

  Points to a buffer that contains an IPv6 routing header.

- index

  Specifies a value that identifies a position in a routing header for a specific address. Valid values range from 0 to the return value from inet6_rth_segments() minus 1.

D.4.2.2 Description

The inet6_rth_getaddr() function uses a specified index value and retrieves a pointer to an address in a Routing header specified by bp. Call inet6_rth_segments() before calling this function in order to determine the number of segments (addresses) in the routing header.

D.4.2.3 Return Values

Upon successful completion, the inet6_rth_getaddr() function returns a pointer to an address.

Upon failure, it returns a NULL pointer.

D.4.3 The inet6_rth_init Call

Initializes an IPv6 routing header buffer.

```
#include <netinet/ip6.h>

void inet6_rth_init( void *bp, int bp_len, int type, int segments );
```

D.4.3.1 Parameters

- bp

  Points to a buffer that is to contain an IPv6 routing header.

- bp_len

  Specifies the length, in bytes, of the buffer.

- type

Specifies the type of routing header. The valid value is IPV6_RTHDR_TYPE_0 for IPv6 routing header type 0.

- segments

  Specifies the number of segments or addresses that are to be included in the routing header. The valid value is from 0 to 127, inclusive.

D.4.3.2 Description

The inet6_rth_init() function initializes a buffer and buffer data for an IPv6 routing header. The function sets the ip60r_segleft, ip6r0_nxt, and ip6r0_reserved members in the ip6_rthdr0 structure to zero. In addition, it sets the ip6r0_type member to type and sets the ip6r0_len member based in the segments parameter. (See RFC 2460 for a description of the actual value.) The ip6_rthdr0 structure is defined in netinet/ip6.h.

The application must allocate the buffer. Use the inet6_rth_space() function to determine the buffer size.

Use the returned pointer as the first argument to the inet6_rth_add() function.

D.4.3.3 Return Values

Upon successful completion, the inet6_rth_init() function returns a pointer to the buffer that is to contain the routing header.

If the type is not supported, the bp is a null, or the number of bp_len is invalid, the function returns a NULL pointer.

D.4.4 The inet6_rth_reverse Call

Reverses the order of addresses in an IPv6 routing header.

```
#include <netinet/ip6.h>

int inet6_rth_reverse( const void *in, void *out );
```

D.4.4.1 Parameters

- in

  Points to a buffer that contains an IPv6 routing header.

- out

  Points to a buffer that is to contain the routing header with the reversed addresses. This parameter can point to the same buffer specified by the in parameter.

D.4.4.2 Description

The inet6_rth_reverse() function reads an IPv6 routing header and writes a new routing header, reversing the order of addresses in the new header. The in and out parameters can point to the same buffer.

The function sets the ip60r_segleft member in the ip6_rthdr0 structure to the number of segments (addresses) in the new header.

The ip6_rthdr0 structure is defined in netinet/ip6.h.

D.4.4.3 Return Values

Upon successful completion, the inet6_rth_reverse() function returns 0.

Upon failure, it returns a -1.

D.4.5 The inet6_rth_segments Call

Returns the number of segments (addresses) in an IPv6 routing header.

```
#include <netinet/ip6.h>

int inet6_rth_segments( const void *bp );
```

Parameters

- bp

  Points to a buffer that contains an IPv6 routing header.

D.4.5.1 Description

The inet6_rth_segments() function returns the number of segments (or addresses) in an IPv6 routing header.

D.4.5.2 Return Values

Upon successful completion, the inet6_rth_segments() function returns the number of segments, 0 (zero) or greater than 0.

Upon failure, it returns a -1.

D.4.6 The inet6_rth_space Call

Returns the number of bytes required for an IPv6 routing header.

```
#include <netinet/ip6.h>

size_t inet6_rth_space( int type, int segments );
```

D.4.6.1 Parameters

- type

  Specifies the type of routing header. The valid value is IPV6_RTHDR_TYPE_0 for IPv6 routing header type 0.

- segments

  Specifies the number of segments or addresses that are to be included in the routing header. The valid value is from 0 to 127, inclusive.

D.4.6.2 Description

The inet6_rth_space() function determines the amount of space, in bytes, required for a routing header. Although the function returns the amount of space required, it does not allocate buffer space. This enables the application to allocate a larger buffer.

If the application uses ancillary data, it must pass the returned length to CMSG_LEN() to determine the amount of memory required for the ancillary data object, including the cmsghdr structure.

Note

If an application wants to send other ancillary data objects, it must specify them to sendmsg() as a single msg_control buffer.

D.4.6.3 Return Values

Upon successful completion, the `inet6_rth_space()` function returns the length, in bytes, of the routing header and the specified number of segments.

If the type is not supported or the number of segments is invalid for the type of routing header, the function returns 0.