

# MultiNet 5.6 Installation and Administrator's Guide

**November 2020**

This guide provides information to configure and manage MultiNet for the experienced system manager. Before using this guide, install and start MultiNet as described in the *MultiNet Installation and Administrator's Guide*.

**Operating System/Version:** OpenVMS VAX V5.5-2 or later

OpenVMS Alpha V6.2 or later

OpenVMS Itanium V8.2 or later

**Software Version:** MultiNet 5.6

**Process Software  
Framingham, Massachusetts  
USA**

The material in this document is for informational purposes only and is subject to change without notice. It should not be construed as a commitment by Process Software. Process Software assumes no responsibility for any errors that may appear in this document.

Use, duplication, or disclosure by the U.S. Government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013.

Third-party software may be included in your distribution of MultiNet, and subject to their software license agreements. See [www.process.com/products/multinet/3rdparty.html](http://www.process.com/products/multinet/3rdparty.html) for complete information.

All other trademarks, service marks, registered trademarks, or registered service marks mentioned in this document are the property of their respective holders.

MultiNet is a registered trademark and Process Software and the Process Software logo are trademarks of Process Software.

Copyright ©2020 Process Software Corporation. All rights reserved. Printed in USA.

If the examples of URLs, domain names, internet addresses, and web sites we use in this documentation reflect any that actually exist, it is not intentional and should not to be considered an endorsement, approval, or recommendation of the actual site, or any products or services located at any such site by Process Software. Any resemblance or duplication is strictly coincidental.

# Preface

This guide is intended for OpenVMS system managers who need to configure and manage MultiNet.

To configure TCP/IP Services for DECnet Applications (formerly known as Phase/IP), refer to the *TCP/IP Services for DECnet Applications Guide* section of the *MultiNet Message, Logicals, and DECnet Applications* book in this MultiNet documentation set.

For information about basic TCP/IP concepts, see the *Installation and Introduction* section of this guide.

## Obtaining Technical Support

Process Software provides technical support if you have a current Maintenance Service Agreement. If you obtained MultiNet from an authorized distributor or partner, you receive your technical support directly from them.

You can contact Technical Support by sending electronic mail or calling the Technical Support center.

## Before Contacting Technical Support

Before you call or send e-mail please verify that your Maintenance Service Agreement is current, and have the following information available:

- Your name
- Your company name
- Your e-mail address
- Your telephone number
- Your Maintenance Agreement Number
- OpenVMS architecture and version
- MultiNet version

Have complete information about your configuration, error messages that appeared, and problem specifics.

Be prepared to let an engineer connect to your system either with TELNET or SSH. Be prepared to give the engineer access to a privileged account to diagnose your problem.

You can obtain information about your OpenVMS architecture, OpenVMS version, and MultiNet version with the `MULTINET SHOW /LICENSE` command. For example:

```
$ MULTINET SHOW /LICENSE  
Process Software MultiNet V5.6, VAXstation 4000-90, OpenVMS VAX V7.1
```

In this example:

- The machine or system architecture is VAX.
- The OpenVMS version is V7.1.
- The MultiNet version is V5.6

## Sending Electronic Mail

For most questions, electronic mail is the preferred communications method. Technical support via electronic mail is available to customers with a current support contract. Send electronic mail to [support@process.com](mailto:support@process.com)

At the beginning of your mail message, include the information listed in the section *Before Contacting Technical Support*. Continue with the description of your situation and problem specifics. Include all relevant information to help your Technical Support Specialist process and track your electronic support request.

Electronic mail is answered Monday through Friday from 9:00 a.m. to 5:00 p.m. United States Eastern Time.

## Calling Technical Support

For regular support issues, call 800-394-8700 or 508-628-5074 for support Monday through Friday from 9:00 a.m. to 5:00 p.m. United States Eastern Time.

For our customers in North America with *critical* problems, an option for support 7 days per week, 24 hours per day is available at an additional charge. Please contact your account representative for further details.

Before calling, have available the information described in the section *Before Contacting Technical Support*. When you call, you will be connected to a Technical Support Specialist.

Be prepared to discuss problem specifics with your Technical Support Specialist and to let that person connect to your system.

If a Specialist is not immediately available, your call will be returned as soon as possible.

# Obtaining Online Help

Extensive information about MultiNet is provided in the MultiNet help library. For more information, use the following command:

```
$ HELP MULTINET
```

## MultiNet Frequently Asked Questions List

You can obtain an updated list of frequently asked questions (FAQs) and answers about MultiNet products from the Process Software home page located at <http://www.process.com/>

## Accessing the MultiNet Public Mailing List

Process Software maintains two public mailing lists for MultiNet customers.

The [Info-MultiNet@lists.process.com](mailto:Info-MultiNet@lists.process.com) mailing list is a forum for discussion among MultiNet system managers and programmers. Questions and problems regarding MultiNet can be posted for a response by any of the subscribers. To subscribe to Info-MultiNet, send a mail message with the word SUBSCRIBE in the body to [Info-MultiNet@lists.process.com](mailto:Info-MultiNet@lists.process.com).

The [MultiNet-Announce@lists.process.com](mailto:MultiNet-Announce@lists.process.com) mailing list is a one-way communication (from Process Software to you) used for the posting of announcements relating to MultiNet (patch releases, product releases, etc.). To subscribe to MultiNet-Announce, send a mail message with the word SUBSCRIBE in the body to [MultiNet-Announce@lists.process.com](mailto:MultiNet-Announce@lists.process.com).

# Obtaining Software Patches Over the Internet

Process Software provides software patches in save set and ZIP format on its anonymous FTP server, ftp.multinet.process.com. For the location of software patches, read the .WELCOME file in the top-level anonymous directory. This file refers you to the directories containing software patches.

To retrieve a software patch, enter the following commands:

```
$ MULTINET FTP /USERNAME-ANONYMOUS /PASSWORD=email FTP.MULTINET.PROCESS.COM
```

A message welcoming you to the Process Software FTP directory appears next followed by the FTP prompt. Enter the following at the prompts:

```
FTP>CD [PATCHES.MULTINETnnn]  
FTP>GET update filename
```

- *emailaddress* is your e-mail address in the standard *user@host* format.
- *nnn* is the version of MultiNet you want to transfer.
- *update\_filename* is the name of the file you want to transfer.

To transfer files from Process Software directly to an OpenVMS system, you can use the GET command without any other FTP commands. However, if you need to transfer a software patch through an intermediate non-OpenVMS system, use BINARY mode to transfer the files to and from that system.

In addition, if you are fetching the software patch in save set format, make sure the save set record size is 2048 bytes when you transfer the file from the intermediate system to your OpenVMS system:

- If you use the GET command to download the file from the intermediate system, use the FTP RECORD-SIZE 2048 command *before* transferring the file.
- If you use the PUT command to upload the file to your OpenVMS system, log into the intermediate system and use the FTP quote site rms recsize 2048 command *before* transferring the file.

The following example shows how to use the UNZIP utility, assuming you have copied the appropriate version of UNZIP.EXE to your current default directory.

```
$ UNZIP := $SYS$DISK:[ ]UNZIP.EXE  
$ UNZIP filename.ZIP
```

Use VMSINSTALL to upgrade your MultiNet system with the software patch.

# Typographical Conventions

Examples in this guide use the following conventions:

Convention	Meaning
host	Any computer system on the network. The local host is your computer. A remote host is any other computer.
monospaced type	<p>System output or user input. User input is in <b>reversed bold</b> type.</p> <p>Example: Is this configuration correct? <b>YES</b></p> <p>Monospaced type also indicates user input where the case of the entry should be preserved.</p>
<i>italic type</i>	Variable value in commands and examples. For example, <i>username</i> indicates that you must substitute your actual username. Italic text also identifies documentation references.
[ <i>directory</i> ]	Directory name in an OpenVMS file specification. Include the brackets in the specification.
[ <i>optional-text</i> ]	<p>(Italicized text and square brackets) Enclosed information is optional. Do not include the brackets when entering the information.</p> <p>Example: START/IP <i>line address</i> [<i>info</i>]</p> <p>This command indicates that the <i>info</i> parameter is optional.</p>
{value   value}	Denotes that you should use only one of the given values. Do not include the braces or vertical bars when entering the value.
<b>Note</b>	Information that follows is particularly noteworthy.

<b>Caution</b>	Information that follows is critical in preventing a system interruption or security breach.
<b>key</b>	Press the specified key on your keyboard.
<b>Ctrl+key</b>	Press the control key and the other specified key simultaneously.
<b>Return</b>	Press the Return or Enter key on your keyboard.



# 1. Installing and Upgrading MultiNet

The below table lists the steps required to install MultiNet. As indicated by the footnotes, you need only perform some of the steps if you are installing this version of MultiNet for the *first time*; you can skip some steps if you have already installed this version of MultiNet on one VMSccluster node of the same architecture (VAX, Alpha, or Itanium). The sections in this chapter provide detailed information about each step.

Step	Description
1	Gather Information for the Installation
2	Read the Release Notes
3	Check OpenVMS and MultiNet Versions
4	Use the Correct Media
5	Back Up Your System Disk
6	Reserve Sufficient Disk Space
7	Log on as SYSTEM
8	Ask Other Users to Log Off
9	Update System Parameters
10	Check the Location of the DCLTABLES . EXE File
11	Review the MultiNet Directory Layout

12	Load the PAK
13	Run VMSINSTALL
14	Establish an Initial Configuration
15	Configure IP Transport
16	Start the New Version of MultiNet
17	Configure Services
18	Add and Update User Exits
19	Install MultiNet Commands in the DCLTABLES . EXE File

Configuration procedures allow you to configure IP connectivity using one LAN interface.

However, you can upgrade MultiNet without changing your existing configuration. As a result, if you upgrade MultiNet on one node in a VMScluster environment, you only need to reboot other nodes of the same architecture to have the upgrade take effect.

## Step 1: Gather Information for the Installation

During installation, you have the option of configuring IP connectivity for the first network device the configuration procedure finds. If you plan to perform this initial configuration, gather the required information before running VMSINSTALL. The below table lists the information you must gather.

If the default configuration is not appropriate for your system, you can configure MultiNet after completing the installation.

**Note:** If you are upgrading a system already running MultiNet, you do not have to configure anything unless you want to change the existing configuration.

Parameter Name	Description
Internet host name	<p>The name by which your system will be known. If you plan to use DNS to resolve host names, you must use the fully qualified host name supplied by your network administrator or Internet access provider (for example, <code>BIGBOOTE.EXAMPLE.COM</code>). It is generally a good idea to use the well-known name for this host; other systems will rely on DNS or host tables to resolve your host name to the appropriate IP address.</p>
IP address	<p>The dotted-decimal representation of your system's IP address. For example, the address 10.1.2.3 is a class B IP address in which 10.1 is the network number and 2.3 is the host number. Obtain your IP address from your network administrator or Internet access provider.</p>
Subnet mask	<p>The dotted-decimal representation of a 32-bit mask that determines the network portion of your IP address to allow your network to be subdivided into multiple network segments.</p> <p>For example: If your IP address is 10.1.2.3, and your subnet mask is 255.255.255.0, your network number is 10.1.2 instead of 2.3 (the network number implied by the class B IP address), and your host number is 3 instead of 2.3. Obtain your subnet mask from your network administrator or Internet service provider.</p>
Default route	<p>The dotted-decimal representation of the IP address of the router to which IP packets are sent when there is no route to the destination host or network in your system's routing table. The default route must have the same network number as your system. For example: If your IP address is 10.1.2.3, and your subnet mask is 255.255.255.0, 10.1.2.126 can be a valid default route, but 10.1.128.126 (on a different subnet) cannot be. Obtain the default route from your network administrator or Internet access provider.</p>
Use DNS	<p>The answer to the question, "Does your system have access to the Internet to take advantage of DNS (Domain Name System) to resolve host names to IP addresses?". Ask your network administrator or Internet service provider. If your system does not have access to the Internet, you will not be able to resolve host names after completing the installation until you configure the DNS server or host tables on your system.</p>

Timezone	The standard abbreviation for your local time zone (For example: EST, CST, MST, or PST). Enter your local time zone, or, if your system clock is not in the local time zone, enter the time zone your system clock uses. When prompted for your time zone, type a question mark (?) to see a list of valid time zone abbreviations. Switching between Standard Time and Daylight Savings Time occurs automatically.
----------	---

## Step 2: Read the Release Notes

The *Release Notes* contain important information about this release that may not be published in this guide or in the other publications in the MultiNet documentation set. You can display or print the *Release Notes* during the installation.

After the installation, you can find the Release Notes in the file  
`SYSSHELP:MULTINET056.RELEASE_NOTES`.

The same information is also available from the Process Software web site at [www.process.com](http://www.process.com) and on CD-ROM distributions: (`[MULTINET056]MULTINET_RELEASE_NOTES.TXT`).

## Step 3: Check OpenVMS and MultiNet Versions

Ensure your system is running VAX/VMS V5.5-2 or later, OpenVMS Alpha V6.2 or later, or OpenVMS Itanium V8.2 or later. If you are not running one of these operating system versions, you *must* upgrade before installing MultiNet.

If you are upgrading from an earlier release of MultiNet, ensure the existing version of MultiNet is V4.3 or later; you cannot upgrade MultiNet directly from V4.2 or earlier versions.

To check your current MultiNet version:

```
$ TYPE MULTINET:MULTINET_VERSION.
```

If you are running a version of MultiNet earlier than V4.3, refer to your old *MultiNet Installation and Administrator's Guides* for instructions on removing MultiNet from your system. Then, install this release of MultiNet as a new installation, *not an upgrade*.

## Step 4: Use the Correct Media

Ensure you have the proper distribution medium. MultiNet is distributed on a CD-ROM that contains the distributions for VAX, Alpha, and Itanium.

**Note:** If you have already installed on another VMScluster node of the same architecture, you do not need to install from the distribution medium again.

## Step 5: Back up Your System Disk

Make a backup copy of your system disk using the OpenVMS BACKUP (or standalone BACKUP) utility.

## Step 6: Reserve Sufficient Disk Space

If you are installing MultiNet for the first time, ensure you have sufficient disk space.

To find out how much free space is available on your system disk, use the following DCL command:

```
$ SHOW DEVICE SYS$SYSDEVICE
```

The information displayed includes the number of free blocks on the disk. The disk space requirements for VAX, Alpha and Itanium versions of MultiNet are available in the MultiNet V5.6 Release Notes.

## Step 7: Log on as SYSTEM

The installation procedure copies files onto the system disk (or another disk you specify). You must be logged in as user name SYSTEM (or as another fully privileged user) to perform the installation.

## Step 8: Ask Other Users to Log Off

Make sure other users log off the system before you start the installation or before you modify any system parameters. You may need to reboot.

## Step 9: Update System Parameters

Before installing MultiNet, you may need to update various system parameters.

1. If you are installing MultiNet on your system for the first time, modify SYS\$SYSTEM:MODPARAMS.DAT as follows:

On VAX systems, add the following lines to SYS\$SYSTEM:MODPARAMS.DAT:

```
MIN_INTSTKPAGES = 10
MIN_CHANNELCNT  = 200
ADD_SPTREQ      = 1700
ADD_GBLSECTIONS = 20
ADD_GBLPAGES    = 340
```

On OpenVMS Alpha systems, add the following lines to `SYS$SYSTEM:MODPARAMS.DAT`:

```
MIN_CHANNELCNT  = 200
ADD_GBLSECTIONS = 20
ADD_GBLPAGES    = 1550
```

On OpenVMS Itanium systems, add the following lines to `SYS$SYSTEM:MODPARAMS.DAT`:

```
MIN_GH_EXEC_DATA = 1496
MIN_GH_EXEC_CODE = 4183
```

2. Execute `SYS$UPDATE:AUTOGEN.COM` to generate a new system parameter file, and reboot your system to activate the new values. Refer to your OpenVMS system management documentation for information about `AUTOGEN`.

```
$ MCR SYSMAN
SYSMAN>SET ENVIRONMENT /CLUSTER
SYSMAN>SET PROFILE /PRIVILEGE-CMKRNL
SYSMAN>PARAMETER USE ACTIVE
SYSMAN>PARAMETER SET LGI_CALLOUTS 0
SYSMAN>PARAMETER WRITE ACTIVE
SYSMAN>EXIT
```

**Note:** If you are using host tables instead of DNS, to ensure they load properly, check the size of the `NETWORK_DATABASE.` and `HOSTTBLUK.DAT` files, and increase the `ADD_GBLPAGES` statement value by 1 for each disk block used by these files, and increase the `ADD_GBLSECTIONS` value by 2, one for each file.

## Step 10: Check the Location of the DCLTABLES.EXE File

Before installing MultiNet, ensure the `DCLTABLES.EXE` file to which you want the `MULTINET` verb added resides in the `SYS$COMMON:[SYSLIB]` directory. `VMSINSTAL` requires you to do this because it will not update copies of `DCLTABLES.EXE` in a system-specific directory.

# Step 11: Review the MultiNet Directory Layout

Unless you specify otherwise, the MultiNet installation procedure creates a top-level [MULTINET] directory on your system disk to hold all the MultiNet files. Depending on the platform on which you are installing, a [.AXP\_COMMON], [.VAX\_COMMON] or [.I64\_COMMON] subdirectory is created in the [MULTINET] directory. As MultiNet is installed on each node, a node- or system-specific directory is also created in the [MULTINET] directory. System-specific MultiNet files are stored in the node-specific directories, and architecture-common files are stored in either the [.AXP\_COMMON], [.VAX\_COMMON] or [.I64\_COMMON] subdirectory.

If you are upgrading an existing directory structure, that structure is retained.

While each of [MULTINET.AXP\_COMMON], [MULTINET.VAX\_COMMON] and [MULTINET.I64\_COMMON] can exist in a multi-architecture cluster in the same directory structure, common configuration files will only be shared by similar architecture nodes.

Unless you specify otherwise, the system-specific directory name is determined by the first of the following values that exist:

- The SYSGEN parameter SCSNODE
- The logical name SYS\$NODE
- The logical name SYS\$TOPSYS

The system-specific directory is created in the [MULTINET] top-level directory.

Two directories are created under each system-specific directory:

- MULTINET.DIR - Contains node-specific files.
- SYSCOMMON.DIR - A file entry that points to the architecture-specific common directory. (Subsequent node-specific directories are created similarly.)

MultiNet relies on the logical name MULTINET\_COMMON\_ROOT to locate shared files, and on the logical name MULTINET\_ROOT to locate system-specific files. MULTINET\_ROOT is a search list of the system-specific directory and the MULTINET\_COMMON\_ROOT directory. The MultiNet startup command procedure, START\_MULTINET.COM, automatically defines these logical names.

## The MultiNet Directory Structure:

```
[MULTINET]
  [.AXP_COMMON]      <--+
    [.MULTINET]      |
  [.SEUSS]           |
    [.MULTINET]      |
    [.SYSCOMMON]     ----+ This is a pointer to [MULTINET.AXP_COMMON]
```

```

[.TOMMY]          |
  [.MULTINET]     |
    [.SYSCOMMON]  ---+ This is also a pointer to [MULTINET.AXP_COMMON]
[.I64_COMMON]    <---+
  [.MULTINET]     |
[.JLS]           |
  [.MULTINET]     |
    [.SYSCOMMON]  ---+ This is a pointer to [MULTINET.I64_COMMON]
[.VAX_COMMON]    <---+
  [.MULTINET]     |
[.ZANE]          |
  [.MULTINET]     |
    [.SYSCOMMON]  ---+ This is a pointer to [MULTINET.VAX_COMMON]

```

## Step 12: Load the PAK (Product Authorization Key)

MultiNet is licensed by a single PAK. You must register and load the PAK if this is the first time you've installed MultiNet on your system.

For more information about PAKs, refer to the *VMS License Management Utility Manual*. See Chapter 2 of this manual for an example of registering and loading a PAK.

To register your new PAK:

1. Start the VMSLICENSE utility with the command:

```
$ @SYS$UPDATE:VMSLICENSE
```

2. Register your new PAK. Answer YES when prompted whether you want it loaded.
3. Exit the VMSLICENSE utility when you have registered your new PAK.

## Step 13: Run VMSINSTAL

Before beginning the installation, gather the information described in the section *Gathering Information for the Installation*.

**Note:** If you are installing MultiNet on one node in a homogeneous VMScluster environment, and have already installed MultiNet on another node of the same architecture, you do not need to run VMSINSTAL; all required files are already in place.



In this procedure, default values appear in square brackets ([default]). To accept the default value, press **RETURN**. To abort the installation at any time, press **Ctrl+Y**.

1. Load your distribution media.

2. Start the VMSINSTAL utility:

```
$ @SYS$UPDATE:VMSINSTAL product loc [OPTIONS N,AWD=device: [directory]]
```

*product* is the name of the MultiNet product. For VAX and Alpha, the name is MULTINET056. For Itanium, the name is MULTINET\_I64056.

*loc* is the device/directory where the installation kit is stored. If you are installing MultiNet from CD-ROM, then mount the CD and specify the full path name of the installation save set directory (such as DKB0:[MULTINET056]). If you are installing MultiNet from your login directory, use the logical name SYS\$LOGIN:.

Use the OPTIONS N parameters if you want to be prompted to view or print the Release Notes.

Use the OPTIONS AWD parameters if there is insufficient space on your system disk to accommodate the temporary files created during installation. See the *VMS System Manager's Manual* for more details about AWD.

Messages similar to the following will appear. The exact wording depends on your system's operating system and architecture:

```
OpenVMS AXP Software Product Installation Procedure V8.2  
It is 08-MAY-2019 at 11:10.  
Enter a question mark (?) at any time for help.
```

3. Ensure your system is in the proper state. If you are not logged in as SYSTEM (or as another fully privileged user), or if any user processes are still running, you are alerted before being asked if you want to continue.

**Note:** Do not continue the installation until you are logged in using a fully-privileged user name such as SYSTEM, and all other users have logged off. You do not, however, need to disable DECnet to install MultiNet.

Any such messages are followed by the prompt:

```
* Do you want to continue anyway [NO]?
```

When you are ready, enter YES. If you enter NO, VMSINSTAL will abort the installation.

4. Make sure you have a reliable copy of your system disk. The installation copies files onto your system disk. The following prompt appears:

```
* Are you satisfied with the backup of your system disk [YES]?
```

If your system disk has not been backed up and you would like to back it up now, enter NO. The installation procedure terminates. After you have made a backup copy of your system disk, restart the installation procedure.

If your system disk is already backed-up, press **RETURN** or enter YES.

5. Indicate whether the media is ready, if necessary.

If you are installing from CD-ROM, the installation begins immediately. Load the distribution media into the drive, and enter YES when you are ready to continue. The following message appears:

```
%MOUNT-I-MOUNTED, MLTNET mounted on _location:  
VMSINSTAL then begins the MultiNet installation.
```

6. Process the Release Notes, if necessary. If you specified the parameter `OPTIONS N` when you started VMSINSTAL in Step 2, you are asked if you want to print or display the Release Notes. Respond as desired. For an example of printing the Release Notes, see the section *Sample Installation* in Chapter 2.

7. Read the Terms and Conditions displayed in the MultiNet Restricted Rights notice.

8. Ensure proper specification of the MultiNet directory structure. If you are upgrading a running version of MultiNet, the logical names `MULTINET_ROOT` and `MULTINET_COMMON_ROOT` will be displayed for verification:

```
The logicals MULTINET_ROOT and MULTINET_COMMON_ROOT are already defined,  
with the following values:
```

```
disk:[MULTINET.nodename.] [MULTINET]  
disk:[MULTINET.nodename.SYSCOMMON] [MULTINET]
```

You are then asked whether you want to upgrade:

```
You may either upgrade the version of MultiNet installed in this location,  
or install a fresh copy of MultiNet in another location.
```

```
* Do you want to upgrade your MultiNet installation [YES] ?
```

If you reply NO, or if this is a new installation, you are prompted for an installation location and a name for the system-specific directory:

```
* Where do you want to install MultiNet [SYS$SYSDEVICE:[MULTINET]]:
```

```
* What do you want to call the system-specific directory [AXP01]:
```

**Caution!** If you respond NO, and install MultiNet in a new location, your new installation will not contain any of your original MultiNet configuration files.

Several lines of copyright notices appear, then you are prompted for each component installation.

9. Specify the software components you want to install.

Enter YES for each software component you want to install, and NO for each one you do not want on your system (previously installed components will be removed). Include all MultiNet products you want to run on any node in your VMScluster environment.

10. Verify your selections. VMSINSTAL lists the components to be installed and the software components, if any, to be removed from your system during installation. You are asked if you want to revise the list of software components to be installed or removed:

```
* Would you like to change your selections [NO]?
```

To change the list of software components to install, enter YES, and repeat Step 9. To accept the list, enter NO.

11. Decide whether to install user commands:

```
* Do you want to install the user commands in DCLTABLES [YES]?
```

This prompt lets you choose whether or not to install additional MultiNet DCL commands in the DCLTABLES . EXE file. Doing this allows you to issue the following commands without the MULTINET prefix.

If you are installing MultiNet for evaluation, or if you have commands with these same names from another vendor, enter NO. You can still access the MultiNet utilities by preceding the command with the keyword MULTINET, or by creating symbols such as:

```
$ TELNET := MULTINET TELNET
```

Otherwise, enter YES to install the user commands in the DCLTABLES . EXE file.

**Note:** You can install the MultiNet user commands in the DCLTABLES . EXE file after completing the installation.

If you have not installed your MultiNet license PAK, you are warned that you do not have the required license. You can still continue to install MultiNet. To continue the process, enter YES at the following prompt:

```
* Do you want to continue the installation anyway [NO]?
```

You must register the PAK before you can start and use MultiNet. When you finish the installation and configuration, run VMSLICENSE to register the PAK as shown in the section *Load the PAK*.

12. Decide whether to purge files:

```
* Do you want to purge files replaced by this installation [YES]?
```

If you want VMSINSTAL to purge files replaced by this version of the software, press **RETURN** or enter YES. If you do not want the replaced files to be purged, enter NO.

**Note:** Even if you choose to purge files, the installation procedure does not purge configuration files.

**CAUTION!** If you upgrade MultiNet in a VMSccluster environment, do not purge the old files during installation. Instead, wait until you have rebooted each node in the cluster.

13. Decide whether to configure software components. If you are installing the TCP/IP applications, you are asked if you want to configure those software components after installation. Enter YES if you are installing MultiNet for the first time, or if you need to reconfigure.

The installation continues with no additional prompts.

If this is the first time you have installed MultiNet, the installation procedure creates the MultiNet directories and copies the files from the distribution kit into those directories.

VMSINSTAL also optionally installs the MultiNet user commands in `SYS$LIBRARY:DCLTABLES.EXE`. The `MULTINET` verb is always installed in your `DCLTABLES.EXE` file.

If you have previously installed MultiNet on your system, the installation procedure deletes old copies of files replaced during this installation. A series of `%VMSINSTAL-I-` and `%MULTINET-I-` informational messages appear during these operations to indicate which files are being installed, merged, or removed, and the save sets from which they came.

If you chose to configure the TCP/IP applications, respond to the prompts with the information you gathered in Step 1.

The configuration procedure then compiles the ASCII network, host, and service configuration files into binary format and creates or updates various configuration files. As it performs these operations, it prints a number of informational messages.

Upon completion, the installation procedure prints messages similar to the following (the messages may vary with the operating system version running on your system):

```
Installation of MULTINET V5.6 completed at 12:15
Adding history entry in VMI$ROOT:[SYSUPD]VMSINSTAL.HISTORY
Creating installation data file: VMI$ROOT:[SYSUPD]MULTINET056.VMI_DATA
VMSINSTAL procedure done at 12:15
$
```

MultiNet is now installed.

## Step 14: Establish an Initial Configuration

When you upgrade from an earlier version of MultiNet, all existing configuration settings are preserved unless you intentionally change them during installation.

If you are installing MultiNet for the first time, however, you have the opportunity during installation to configure the MultiNet IP transport over one standard network interface simply by responding to a series of prompts.

If you have already installed MultiNet on one VMSccluster node and are now installing MultiNet on another VMSccluster node of the same architecture and operating system, or if you chose not to configure MultiNet during the installation, you can configure the MultiNet IP transport over the standard interface using the `CONFIGURE.COM` command procedure. Gather the required configuration information before configuring. Step 1 describes the information you need to gather.

**CAUTION!** If your system does not have one of the standard network interfaces, do not use `CONFIGURE.COM` to establish connectivity. If you do not use the command procedure, you will need to manually enter configuration information. For details on establishing connectivity with all supported network interfaces, refer to the appropriate chapter in this document.

## Step 15: Configure IP Transport over the Standard Network Interface

Follow these steps to configure the MultiNet IP transport over the standard network interface:

1. Set your default directory to the architecture-specific common directory, `device: [MULTINET.arch_COMMON.MULTINET]. device` is the device you chose in Step 8 of the installation procedure, and `arch` is the architecture (either VAX, AXP, or Itanium).
2. The MultiNet configuration command procedure `CONFIGURE.COM` provides a default location for the new node-specific directories and prompts you for correction. It will create the new directories and logical names to get your system up and running on the local subnet by prompting you for the information in your configuration checklist (see Step 1 of this installation procedure). To run the command procedure, type:

```
$ @CONFIGURE
```

## Step 16: Start the New Version of MultiNet

If you installed MultiNet on a system already running MultiNet, perform these four steps after completing the installation to ensure its success; MultiNet then starts automatically:

1. Recompile the host lookup table (recompiling the host lookup table is only necessary on the first node of a given architecture in a VMScluster environment):

```
$ MULTINET_HOST_TABLE_COMPILE
```

2. Reboot your system to load the new kernel.

3. Run the MultiNet CHECK utility.

```
$ MULTINET_CHECK
```

4. Rectify any problems found.

If you have successfully installed and configured MultiNet for the first time on your system, as described in the previous sections, you are ready to start MultiNet. You do not have to reboot if your system has never run any TCP/IP stack; however, you may need to reboot after running AUTOGEN to activate any system parameter changes you made.

Successful configuration of MultiNet defines the logical name MULTINET and creates the file MULTINET:START\_MULTINET.COM. However, if you have rebooted, the MULTINET logical may not be defined. Ensure the logical name is defined. Enter:

```
$ SHOW_LOGICAL_MULTINET
```

Also ensure the file exists. Enter:

```
$ DIRECTORY_MULTINET:START_MULTINET.COM
```

If the logical name and the file are not present, check for installation errors and configuration problems, and correct them before proceeding. If they are not present because you rebooted, then add the following line to your system startup command procedure:

```
$ @SYS$SYSDEVICE:[MULTINET.BIGBOOTE.MULTINET]START_MULTINET
```

To locate the startup command procedure, enter the following:

```
$ DIR SYS$SYSDEVICE:[000000... ]START_MULTINET.COM
Directory SYS$SYSDEVICE:[MULTINET.BIGBOOTE.MULTINET]
START_MULTINET.COM;1
```

Use the following commands to start MultiNet:

```
$ REPLY/ENABLE=NETWORK/TEMPORARY
$ @MULTINET:START_MULTINET
```

You may use the MULTINET logical name here, as it is defined by the configuration procedures. Be sure your system startup command procedure refers to the disk and directory where you installed MultiNet.

**Note:** Regardless of where you installed MultiNet, you do not have to define the `MULTINET` logical name - this is done automatically in the first lines of the `START_MULTINET.COM` command procedure.

## Step 17: Configure Services

Once you have established IP connectivity, you can configure services and other components. Documentation on configuring standard services are contained in this document.

## Step 18: Add and Update User Exits

MultiNet allows you to customize some functions through the use of user exits. If you are upgrading MultiNet and have modified any user exits prior to this installation, merge your modifications into the user exits replaced during the installation. For more information about customizing user exits, see the *MultiNet Installation and Administrator's Guide*.

## Step 19: Install MultiNet Commands in the DCLTABLES.EXE File

If you did not install the MultiNet user commands in the `DCLTABLES.EXE` file during installation, you can install them manually with the following commands:

```
$ SET COMMAND /TABLES=SYS$COMMON:[SYSLIB]DCLTABLES -  
$ /OUTPUT=SYS$COMMON:[SYSLIB]DCLTABLES MULTINET:USER.CLD  
$ INSTALL REPLACE SYS$COMMON:[SYSLIB]DCLTABLES
```

## Removing MultiNet

You can use the de-installation command procedure to remove MultiNet. The procedure removes all MultiNet files from disk and all MultiNet commands from the `DCLTABLES.EXE` file. Before executing the procedure:



1. Use the `LICENSE DISABLE` or `DELETE` commands, or the `VMSLICENSE` command procedure to remove any registered MultiNet PAKs (Product Authorization Keys). Refer to the *VMS License Management Utility Manual* for more information about PAKs.
2. Remove the reference to `START_MULTINET.COM` from your system startup command procedure.
3. Remove system parameter changes in `MODPARAMS.DAT` made as part of the installation; invoke the `AUTOGEN` utility to set the parameters back to their pre-installation values. Refer to the *Guide to Maintaining a VMS System* for descriptions of `MODPARAMS.DAT` and `AUTOGEN` parameters.
4. Reboot your system (or systems, if you installed MultiNet on more than one node in a VMScluster environment).

To execute the MultiNet de-installation procedure, type this command if MultiNet is still running:

```
$ @MULTINET:REMOVE
```

Otherwise, type:

```
$ @device:[directory.node.SYSCOMMON.MULTINET]REMOVE
```

*device*, *directory*, and *node* are those you specified during Step 8 of the installation procedure.

# 2. Example Procedures

This chapter contains example procedures that show the prompts you encounter when registering the PAK and during the installation procedure. The user responses, which appear in **reversed bold** typeface, are provided only to illustrate how you may respond. *Do not use these responses for your installation!* Use the configuration information you gathered in Step 1 of Chapter 1.

## Installing a License PAK

This section presents an example dialog showing how to use the `VMSLICENSE REGISTER` option with a PAK (Product Authorization Key).

The values shown below are only examples. When you use the `VMSLICENSE REGISTER` option, use the actual values provided with your PAK.

**Note:** Refer to the section in Chapter 1 titled *Load the PAK (Product Authorization Key)* for the procedure to use for registering the PAK on software upgrades.

Refer to the *VMS License Management Utility Manual* for more information about PAKs.

### Registering and Loading a PAK:

```
$ @SYS$UPDATE:VMSLICENSE
VMS License Management Utility Options:
1. REGISTER a Product Authorization Key
2. AMEND an existing Product Authorization Key
3. CANCEL an existing Product Authorization Key
4. LIST the Product Authorization Keys
5. MODIFY an existing Product Authorization Key
6. DISABLE an existing Product Authorization Key
7. DELETE an existing Product Authorization Key
8. COPY an existing Product Authorization Key
9. MOVE an existing Product Authorization Key
10. ENABLE an existing Product Authorization Key
```

11. SHOW the licenses loaded on this node
  12. SHOW the unit requirements for this node
99. EXIT this procedure

Type '?' at any prompt for a description of the information requested.  
Press Ctrl/Z at any prompt to exit this procedure.  
Enter one of the above choices [1] **1**  
Do you have your Product Authorization Key? [YES] **RETURN**

Use the REGISTER option to add a new license to a license database. A Product Authorization Key (PAK) provides the product name and information you need to register the license. You must enter all the information provided by your PAK exactly as it appears.

Issuer [DEC]: **PROCESS SOFTWARE**  
Authorization Number []: **B-400-17904**  
Product Name [ ]: **MULTINET**  
Producer [DEC]: **PROCESS SOFTWARE**  
Number of Units []: **100**  
Version []: **RETURN**  
Product Release Date []: **23-MAR-2020**  
Key Termination Date []: **RETURN**  
Availability Table Code []: **F**  
Activity Table Code []: **RETURN**  
Key Options []: **NO SHARE**

This Product Authorization Key (PAK) has been provided with the NO\_SHARE option. If it is to be used by a node in a cluster, this PAK must be restricted to a specific node.

If this PAK is to be used by a standalone system, answer NO to the following question.

Is this PAK restricted to a cluster member node? [YES]: **RETURN**

Note: For the majority of systems, the SCS node name is the same as the DECnet node name.

Node this PAK is restricted to (SCS Node name) []: **BIGBOOTE**  
Product Token []: **AB-400-17904**  
Hardware-Id []: **RETURN**  
Checksum []: **4-IPMA-KEIL-PCOP-HNNJ**

Here is a list of the license information just entered:

Issuer: PROCESS SOFTWARE  
Authorization: B-400-17904  
Producer: PROCESS SOFTWARE  
Units: 100  
Release Date: 23-MAR-2020  
Version:

```
Termination Date:
Availability:  F
Activity:
Options:  NO_SHARE
Token:  AB-400-17904
Hardware ID:
Checksum:  4-IPMA-KEIL-PCOP-HNNJ
```

This authorization key is restricted to: BIGBOOTE

Is that correct? [YES] **RETURN**

Do you want to LOAD this license on this system? [YES] **RETURN**

VMS License Management Utility Options:

1. REGISTER a Product Authorization Key
2. AMEND an existing Product Authorization Key
3. CANCEL an existing Product Authorization Key
4. LIST the Product Authorization Keys
5. MODIFY an existing Product Authorization Key
6. DISABLE an existing Product Authorization Key
7. DELETE an existing Product Authorization Key
8. COPY an existing Product Authorization Key
9. MOVE an existing Product Authorization Key
- 10.ENABLE an existing Product Authorization Key
- 11.SHOW the licenses loaded on this node
- 12.SHOW the unit requirements for this node

99.EXIT this procedure

Type '?' at any prompt for a description of the information requested.  
Press Ctrl/Z at any prompt to exit this procedure.

Enter one of the above choices [1] **99**

# Printing the Consolidated Release Notes

The example below shows how to print the *Release Notes*.

**Note:** The VMSINSTALL kit has first been copied to the directory DUA0 : [SWDIST].

```
$ @SYS$UPDATE:VMSINSTAL MULTINET056 DUA0:[SWDIST] OPTIONS N
OpenVMS AXP Software Product Installation Procedure V8.4
It is 08-MAR-2020 at 11:10.
Enter a question mark (?) at any time for help.
* Are you satisfied with the backup of your system disk [YES]? RETURN
The following products will be processed:
MULTINET V5.6
Beginning installation of MULTINET V5.6 at 11:10
%VMSINSTAL-I-RESTORE, Restoring product save set A ...
Release notes included with this kit are always copied to SYS$HELP.
Additional Release Notes Options:
1. Display release notes
2. Print release notes
3. Both 1 and 2
4. None of the above
* Select option [2]: RETURN
* Queue name [SYS$PRINT]: SYS$PRINT
Job MULTINET056 (queue SYS$PRINT, entry 1023) started on LPA0:
* Do you want to continue the installation [NO] ? RETURN
%VMSINSTAL-I-RELMOVED, The product's release notes have been successfully
moved to SYS$HELP.
VMSINSTAL procedure done at 11:11
```

## Sample Installation Dialog

This section contains two examples of MultiNet installation procedures:

- The *Sample New Installation* shows a new installation from CD-ROM. (Some messages may vary with the operating system version.)
- The *Sample Upgrade Installation* shows how to upgrade from an older version of MultiNet to the current release. (Some messages may vary with the operating system version.) You must reboot your system after upgrading the software.

**Note:** The values shown in the dialogs are samples and for illustration purposes only! When you install MultiNet, use the actual values appropriate for your system.

# Sample New Installation

```
$ @SYS$UPDATE:VMSINSTAL MULTINET056 DKB300:[MULTINET056]
```

```
OpenVMS VAX Software Product Installation Procedure V8.4
```

```
It is 2-FEB-2020 at 14:29
```

```
Enter a question mark (?) at any time for help.
```

```
* Are you satisfied with the backup of your system disk [YES] ? RETURN
```

```
The following products will be processed:
```

```
MULTINET V5.6
```

```
Beginning installation of MULTINET V5.6 at 14:30
```

```
%VMSINSTAL-I-RESTORE, Restoring product save set A ...
```

```
%VMSINSTAL-I-REMOVED, Product's release notes have been moved to SYS$HELP.
```

```
* Where do you want to install MultiNet [SYS$SYSDEVICE:[MULTINET]]:RETURN
```

```
* What do you want to call the system-specific directory [HOBBS]: RETURN
```

```
MultiNet (R)
```

```
ALL RIGHTS RESERVED USER THE COPYRIGHT LAWS OF THE UNITED STATES
```

This licensed material is the valuable property of Process Software. Its use, duplication, or disclosure is subject to the restrictions set forth in the License Agreement.

Other use, duplication or disclosure, unless expressly provided for in the license agreement, is unlawful.

```
Installing MultiNet V5.6 Rev A
```

```
* Do you want to install the TCP/IP applications [YES]? RETURN
```

```
* Do you want to install the SSH software [YES]? RETURN
```

```
* Do you want to install the NFS Client software [YES]? RETURN
```

```
* Do you want to install the NFS Server software [YES]? RETURN
```

```
* Do you want to install the online documentation [YES]? RETURN
```

```
The PDF documentation requires 30,000 blocks.
```

```
* Do you want to install the include and library files [YES]? RETURN
```

```
The MultiNet base networking software will be installed with these selected components:
```

```
* TCP/IP applications
```

```
* SSH
```

```
* NFS client
```

```
* NFS server
```

```
* Online documentation
```

```
* Include and library files
```

```
* Would you like to change your selection [NO]? RETURN
```

```
* Do you want to install the user commands in DCLTABLES [YES]? RETURN
```

```
* Do you want to purge files replaced by this installation [YES]? RETURN
```

```
* Configure MultiNet TCP/IP after installation [NO]? YES
```

```
%VMSINSTAL-I-...This product creates...MU$SPECIFIC_ROOT:[MULTINET].
```

```
%VMSINSTAL-I-...This product creates...MU$SPECIFIC_ROOT:[MULTINET.SPOOL].
```

```
%VMSINSTAL-I-...This product creates...MU$SPECIFIC_ROOT:[MULTINET.HELP].
```

```
%VMSINSTAL-I-...This product creates...
```

```
MU$SPECIFIC_ROOT:[MULTINET.LOADABLE_IMAGES].
```

```
%VMSINSTAL-I-...This product creates...MU$COMMON_ROOT:[MULTINET].
```

```
%VMSINSTAL-I-...This product creates...MU$COMMON_ROOT:[MULTINET.SPOOL].
```

```
%VMSINSTAL-I-...This product creates...MU$COMMON_ROOT:[MULTINET.HELP].
```

```
%VMSINSTAL-I-...This product creates... MU$COMMON_ROOT:[MULTINET.LOADABLE_IMAGES].
```

```
The installation will now proceed with no further questions.
```

```
%VMSINSTAL-I-RESTORE, Restoring product save set B...
```

```

%VMSINSTAL-I-RESTORE, Restoring product save set C...
%MULTINET-I-INSTALLING, Installing MultiNet base files
%VMSINSTAL-I-RESTORE, Restoring product save set E...
%MULTINET-I-INSTALLING, Installing MultiNet driver files
%VMSINSTAL-I-RESTORE, Restoring product save set G...
%VMSINSTAL-I-RESTORE, Restoring product save set H...
%MULTINET-I-INSTALLING, Installing MultiNet TCP/IP application files
%VMSINSTAL-I-RESTORE, Restoring product save set J...
%VMSINSTAL-I-RESTORE, Restoring product save set K...
%MULTINET-I-INSTALLING, Installing MultiNet NFS files
%VMSINSTAL-I-RESTORE, Restoring product save set M...
%VMSINSTAL-I-RESTORE, Restoring product save set P...
%MULTINET-I-INSTALLING, Installing the online documentation files
%VMSINSTAL-I-...This product creates...MU$SPECIFIC_ROOT:[MULTINET.DECW$BOOK].
%VMSINSTAL-I-...This product creates...MU$COMMON_ROOT:[MULTINET.DECW$BOOK].
%VMSINSTAL-I-RESTORE, Restoring product save set Q...
%VMSINSTAL-I-RESTORE, Restoring product save set R...
%MULTINET-I-INSTALLING, Installing the include and library files
%VMSINSTAL-I-...This product creates...MU$SPECIFIC_ROOT:[MULTINET.INCLUDE].
%VMSINSTAL-I-...This product creates...MU$SPECIFIC_ROOT:[MULTINET.INCLUDE.ARPA].
%VMSINSTAL-I-...This product creates...MU$SPECIFIC_ROOT:[MULTINET.INCLUDE.NET].
%VMSINSTAL-I-...This product creates...
MU$SPECIFIC_ROOT:[MULTINET.INCLUDE.NETINET].
%VMSINSTAL-I-...This product creates...MU$SPECIFIC_ROOT:[MULTINET.INCLUDE.NETNS].
%VMSINSTAL-I-...This product creates...
MU$SPECIFIC_ROOT:[MULTINET.INCLUDE.NETWARE].
%VMSINSTAL-I-...This product creates...MU$SPECIFIC_ROOT:[MULTINET.INCLUDE.NFS].
%VMSINSTAL-I-...This product creates...MU$SPECIFIC_ROOT:[MULTINET.INCLUDE.RPC].
%VMSINSTAL-I-...This product creates...MU$SPECIFIC_ROOT:[MULTINET.INCLUDE.SYS].
%VMSINSTAL-I-...This product creates...MU$SPECIFIC_ROOT:[MULTINET.INCLUDE.VMS].
%VMSINSTAL-I-...This product creates...MU$SPECIFIC_ROOT:[MULTINET.LIBRARY].
%VMSINSTAL-I-...This product creates...MU$SPECIFIC_ROOT:[MULTINET.EXAMPLES].
%VMSINSTAL-I-...This product creates...MU$COMMON_ROOT:[MULTINET.INCLUDE].
%VMSINSTAL-I-...This product creates...MU$COMMON_ROOT:[MULTINET.INCLUDE.ARPA].
%VMSINSTAL-I-...This product creates...MU$COMMON_ROOT:[MULTINET.INCLUDE.NET].
%VMSINSTAL-I-...This product creates...MU$COMMON_ROOT:[MULTINET.INCLUDE.NETINET].
%VMSINSTAL-I-...This product creates...MU$COMMON_ROOT:[MULTINET.INCLUDE.NETNS].
%VMSINSTAL-I-...This product creates...MU$COMMON_ROOT:[MULTINET.INCLUDE.NETWARE].
%VMSINSTAL-I-...This product creates...MU$COMMON_ROOT:[MULTINET.INCLUDE.NFS].
%VMSINSTAL-I-...This product creates...MU$COMMON_ROOT:[MULTINET.INCLUDE.RPC].
%VMSINSTAL-I-...This product creates...MU$COMMON_ROOT:[MULTINET.INCLUDE.SYS].
%VMSINSTAL-I-...This product creates...MU$COMMON_ROOT:[MULTINET.INCLUDE.VMS].
%VMSINSTAL-I-...This product creates...MU$COMMON_ROOT:[MULTINET.LIBRARY].
%VMSINSTAL-I-...This product creates...MU$COMMON_ROOT:[MULTINET.EXAMPLES].

%MULTINET-I-INSTALLING, Installing MultiNet HELP library
%MULTINET-I-DELETING, Deleting obsolete MultiNet files
% VMSINSTAL-I-MOVEFILES, Files will now be moved to their target directories...
* System Specific directory name: [DKA500:[MULTINET.BIGBOOTE]]: RETURN
* Enter your Internet host name: BIGBOOTE.EXAMPLE.COM
* Enter the Internet (IP) Address for interface EWA0: 10.1.2.3
* Enter the Subnet mask for interface EWA0 (optional): 255.255.255.0
Configure other network devices after installation using the $ MULTINET CONFIGURE
utility.
* Enter the Internet (IP) Address of your default route (optional): 10.1.1.126
* Use Domain Nameservice instead of host tables [YES]? RETURN
* Enter your local timezone: EST

```

To have MultiNet start automatically when your system boots, add the following command to your system startup procedure:

```
$ @DISK$HOBBES:[MULTINET.BIGBOOTE.MULTINET]START_MULTINET.COM
```

```
MultiNet Network Configuration Utility V5.6 (102)
[Reading in MAXIMUM configuration from MULTINET:MULTINET.EXE]
[Reading in configuration from MULTINET:NETWORK_DEVICES.CONFIGURATION]
[Writing Startup file MULTINET:START_MULTINET.COM]
```

```
Installation of MultiNet completed at 14:49
Adding history entry in VMI$ROOT:[SYSUPD]MULTINET056.VMI_DATA
Creating installation data file: VMI$ROOT:[SYSUPD]MULTINET056
VMSINSTAL procedure done at 14:50
```

## Sample Upgrade Installation:

```
$ @SYS$UPDATE:VMSINSTAL MULTINET056 DKB400 OPTIONS N
```

```
OpenVMS AXP Software Product Installation Procedure V8.4
It is 2-NOV-2019 at 12:12.
```

Enter a question mark (?) at any time for help.

\* Are you satisfied with the backup of your system disk [YES]? **RETURN**

The following products will be processed:

MULTINET V5.6

Beginning installation of MULTINET V5.6 at 12:12

%VMSINSTAL-I-RESTORE, Restoring product save set A ...

%VMSINSTAL-I-RELMOVED, Product's release notes have been moved to SYS\$HELP.

The logical names MULTINET\_ROOT and MULTINET\_COMMON\_ROOT are already defined, with the following values:

DKA100:[BOS1.] [MULTINET]

DKA100:[BOS1.SYSCOMMON.] [MULTINET]

You may either upgrade the version of MultiNet installed in this location, or install a fresh copy of MultiNet in another location.

\* Do you want to upgrade your MultiNet installation [YES] ? **YES**

MultiNet (R)

ALL RIGHTS RESERVED UNDER THE COPYRIGHT LAWS OF THE UNITED STATES

This licensed material is the valuable property of Process Software. Its use, duplication, or disclosure is subject to the restrictions set forth in the License Agreement.

Other use, duplication or disclosure, unless expressly provided for in the license agreement, is unlawful.

Installing MultiNet V5.6

\* Do you want to install the TCP/IP applications [YES]? **RETURN**

\* Do you want to install the SSH software [YES]? **RETURN**

\* Do you want to install the NFS Client software [YES]? **RETURN**

\* Do you want to install the NFS Server software [YES]? **RETURN**

\* Do you want to install the online documentation [YES]? **RETURN**

\* Do you want to install the include and library files [YES]? **RETURN**

The MultiNet base networking software will be installed with these selected components:

\* TCP/IP applications

\* SSH



```
* NFS Client
* NFS Server
* Online documentation
* Include and library files
* Would you like to change your selections [NO]? RETURN
* Do you want to install the user commands in DCLTABLES [YES]? RETURN
* Do you want to purge files replaced by this installation [YES]? RETURN
* Configure MultiNet TCP/IP after installation [NO]? RETURN
```

The installation will now proceed with no further questions.

```
%VMSINSTAL-I-RESTORE, Restoring product save set B...
%VMSINSTAL-I-RESTORE, Restoring product save set D...
%MULTINET-I-INSTALLING, Installing MultiNet base files
%VMSINSTAL-I-RESTORE, Restoring product save set F...
%MULTINET-I-INSTALLING, Installing MultiNet driver files
%VMSINSTAL-I-RESTORE, Restoring product save set G...
%VMSINSTAL-I-RESTORE, Restoring product save set I...
%MULTINET-I-INSTALLING, Installing MultiNet TCP/IP application files
%VMSINSTAL-I-RESTORE, Restoring product save set J...
%VMSINSTAL-I-RESTORE, Restoring product save set L...
%MULTINET-I-INSTALLING, Installing MultiNet NFS files
%VMSINSTAL-I-RESTORE, Restoring product save set M...
%VMSINSTAL-I-RESTORE, Restoring product save set P...
%MULTINET-I-INSTALLING, Installing the online documentation files
%VMSINSTAL-I-RESTORE, Restoring product save set Q...
%VMSINSTAL-I-RESTORE, Restoring product save set S...
%MULTINET-I-INSTALLING, Installing the include and library files
%MULTINET-I-INSTALLING, Installing the MultiNet HELP library
%MULTINET-I-MERGING, Merging SERVICES.MASTER_SERVER file
%MULTINET-I-DELETING, Deleting obsolete MultiNet files
%VMSINSTALL-I-MOVEFILES, Files will now be moved to their target directories...
```

```
MultiNet Network Configuration Utility V5.6 (102)
[Reading in MAXIMUM configuration from MULTINET:MULTINET.EXE]
[Reading in configuration from MULTINET:NETWORK_DEVICES.CONFIGURATION]
[Writing Startup file MULTINET:START_MULTINET.COM]
Installation of MULTINET V5.6 completed at 12:30
Adding history entry in VMS$ROOT:[SYSUPD]VMSINSTAL.HISTORY
Creating installation data file: VMI$ROOT:[SYSUPD]MULTINET056.VMI_DATA
VMSINSTAL procedure done at 12:32
```

**Note:** Remember to reboot your system after upgrading the software.

# 3. MultiNet Documentation and Online Help

This chapter provides information about the MultiNet documentation set and available online help for MultiNet.

## The MultiNet Documentation Set

The MultiNet documentation set consists of the following publications:

<i>MultiNet Release Notes</i>	Describes new features in the software, known restrictions or problems, and corrections to the published MultiNet documentation. Review the file <code>SYS\$HELP:MULTINET056.RELEASE_NOTES</code> after installing the software. The same information is also available on CD-ROM distributions in ASCII format ( <code>[MULTINET056]MULTINET_RELEASE_NOTES.TXT</code> )
<i>MultiNet User's Guide</i>	Explains how to explore your network, send and receive electronic mail, log into a remote system, transfer files between systems, and use DECwindows with MultiNet. Includes user command references and the MultiNet Master Index.
<i>MultiNet Installation and Administrator's Guide</i>	Explains how to install, configure, and manage MultiNet.
<i>MultiNet Administrator's Reference</i>	Identifies and describes MultiNet configuration and management commands.
<i>MultiNet Messages, Logicals, and TCP/IP Services for DECnet Applications</i>	Lists MultiNet messages and provides trouble-shooting information as well as MultiNet logicals.

	Explains how to configure TCP/IP Services for DECnet Applications.
<i>MultiNet Programmer's Reference</i>	Describes the MultiNet programming interfaces.

## MultiNet Online Help

MultiNet provides an extensive set of online help topics available using the DCL prompt:

```
$ HELP MULTINET
```

# 4. Introduction to MultiNet and TCP/IP Concepts

This chapter presents a brief description of MultiNet and general concepts useful for understanding the MultiNet software and TCP/IP networking.

## What is MultiNet?

MultiNet is a collection of software that conforms to the set of internationally accepted standards for information exchange known as the TCP/IP protocol suite. The MultiNet software permits your VMS system to interact with other systems running TCP/IP software including PCs, Apple Macintosh systems, UNIX systems, and many others. Because TCP/IP is used on the Internet, using MultiNet lets you communicate locally, or globally with millions of other users and information services.

MultiNet provides applications, configuration tools, and programming libraries that make access to TCP/IP understandable and straight-forward. Whether your system serves one user or thousands of users, MultiNet gives all users access to a wide range of features that extend their use of the network and increase their productivity.

MultiNet works with the OpenVMS Operating System on the VAX, Alpha, and Itanium architectures. On the VAX architecture, MultiNet works with OpenVMS 5.5-2 and later. On the Alpha architecture, MultiNet works with OpenVMS AXP 6.2 and later. On the Itanium architecture, MultiNet works with OpenVMS IA64 8.2 and later. MultiNet is distributed on CD-ROM.

## MultiNet for Users

With MultiNet, users can:

- Send electronic mail to and receive electronic mail from other computer systems using SMTP extensions to OpenVMS Mail and ALL-IN-1 mail.
- Access the Internet and other information services.
- Log into remote systems using TELNET, RLOGIN, or SSH.
- Execute commands on remote systems using RSHELL or SSH.
- Transfer files between local and remote systems with FTP, RCP, TFTP, SCP, and SFTP.
- Print files and manage print jobs on remote systems with the LPD and LPRM utilities.

- Talk to other users interactively with the TALK utility.
- Display information about other sites and users with the FINGER, RUSERS, and WHOIS utilities.
- Read online information about MultiNet using the DCL HELP facility.

## MultiNet for System Managers

With MultiNet, system managers can:

- Configure devices and services easily with command line-based configuration utilities.
- Provide IP connectivity for a variety of networking environments including IP-over-DECnet, Ethernet, FDDI, PPP, SLIP, and X.25.
- Provide other networking connectivity over IP, including DECnet-over-IP.
- Provide access to NFS-mounted file systems with the MultiNet NFS software.
- Change the current configuration dynamically by modifying logical name definitions or by using the NETCONTROL utility.
- Provide security for logging into systems across the network with Kerberos and SSH software.
- Create and access name servers with DNS (Domain Name System) software.
- Configure dynamic routing with the GATED service which supports routing protocols such as RIP, BGP, and others.
- Manage remote printing to print servers or to printers connected to the network with the LPD and stream client software.
- Provide remote access to local OpenVMS printers with the LPD server software.
- Provide electronic mail services with the SMTP and POP protocols; MultiNet provides SMTP enhancements for Message Router (MR), OpenVMS Mail, and ALL-IN-1.
- Access local and remote CD-ROMs, DATs, and conventional magnetic tape devices with the RMTALLOC utility.
- Synchronize system clocks from a central time server with NTP software and provide time updates to other hosts on the network.
- Provide binary compatibility with HP TCP/IP Services for OpenVMS (formerly called UCX) to support Hewlett-Packard and third-party applications such as TeamLinks, DECmcc, and applications written to use DCE for OpenVMS.
- Diagnose system problems and messages with the CHECK, PING, TCPDUMP, TCPVIEW, TRACEROUTE, and X11DEBUG utilities.
- View online information using either the DCL HELP facility.
- Access RFCs on the MultiNet CD-ROM consolidated distribution.

# MultiNet for Programmers

With MultiNet, programmers can:

- Program with socket library routines.
- Work with a \$QIO interface.
- Program with RPC library routines.
- Access sample programs and user exits that can be used to provide additional security and to customize other services (such as printing).

# TCP/IP Concepts

This section describes some of the basic concepts of TCP/IP networking.

## Physical Networks

Physical networks are the cables and associated wiring components that link computers to one another for network communications. Common physical networks are Ethernet, Token Ring, FDDI (Fiber Distributed Data Interface), point-to-point links, and telephone with modems.

## LAN (Local Area Network) Hardware Addresses

Network interface board manufacturers assign a unique hardware (physical) address to each interface board they produce. These hardware addresses are burned into the circuit at the time of manufacture, but can usually be overridden later by a network administrator, if desired.

A hardware address is usually composed of six numbers, one for each octet or eight-bit byte in the address value, separated by colons, such as 00:DD:A8:13:48:C5. The first three octets identify the manufacturer, while the remaining three octets are unique to the board.

Hardware addresses identify individual interfaces and aid in fast and efficient delivery of packets on the physical network.

## IP Addresses

IP addresses identify hosts or interfaces on an IP network. An IP address consists of four numbers, one for each octet or eight-bit byte in the address value. IP addresses are written in dotted-decimal format, such as 10.1.2.3.

An IP address has two basic parts:

- A network number
- A host number

Traditionally, the portions of the address that identify the network and host were determined by the class of the network:

Class A networks	Class A addresses are identified by a value from 1 to 127 in the first octet, such as in 26.1.1.1. In class A addresses, the first octet identifies the network, while the three remaining octets identify the host. For example, IP address 26.1.1.1 identifies host 1.1.1 on network 26.
Class B networks	Class B addresses are identified by a value from 128 to 191 in the first octet, such as in 161.1.1.1. In class B addresses, the first and second octets identify the network, while the remaining two octets identify the host. For example, IP address 161.1.1.1 identifies host 1.1 on network 161.1.
Class C networks	Class C addresses are identified by a value from 192 to 223 in the first octet, such as in 197.1.1.1. In class C addresses, the first three octets identify the network, while the remaining octet identifies the host. For example, IP address 197.1.1.1 identifies host 1 on network 197.1.1.

With the introduction of subnet masks, the division between the network and host portions of an IP address has become much more flexible. See *Subnet Masks* for more information.

The network class determines the size of the network. A class A network can have 16,777,214 hosts, while a class B network can have 65,534 hosts, and a class C network can have only 254 hosts.

## Subnet Masks

The original Internet addressing scheme made it possible for every host on a network to talk directly with every other host on the same network; other hosts were directly accessible if they used the same network number. In class A and class B networks, where very large numbers of hosts with the same network number are available, this scheme is no longer realistic because the underlying physical networks are constrained by bandwidth considerations. Ethernet and Token Ring networks cannot accommodate thousands or hundreds of thousands of hosts in a single, flat network space.

*Subnet masks* allow you to create multiple smaller networks from host addresses. For example, a class A network can be partitioned into class C subnetworks. These smaller, internal networks are called subnets. Subnet addresses are not exposed outside of the network; all changes to accommodate the

additional addresses are handled internally. This simplifies routing information for the network and minimizes the amount of information the network must advertise externally.

Inside the network, you determine how to reallocate addresses by choosing how many bits of the host portion of each address are used as the subnet address and how many bits are used as the host address. You use subnet masks to divide the existing addresses into network and host portions. The subnet mask identifies how much of the existing address can be used as the network portion. The underlying physical network must also be divided into smaller, physical subnets when using a subnet mask to create subnets.

The following example illustrates how to create class C subnets from a class B network address:

The class B network address 10.44.0.0 can be divided by reallocating the first 24 bits of the 32-bit IP address to subnet addressing using the netmask 255.255.255.0. This reallocation allows you to use 10.44.1.0, 10.44.2.0, and so forth, up to 10.44.254.0 as network addresses. All traffic bound for any IP address beginning with the 16-bit network portion 161.44 will be routed to your site where internal routers handle subnetwork addresses. Valid addresses on the internal network, such as 10.44.4.42 and 10.44.224.12, can be reached from anywhere on the Internet; final delivery is handled by the routers on the individual physical subnets that contain the hosts associated with those addresses.

## Broadcast Addresses

A system uses broadcast addresses to send information to all hosts on the network. Packets addressed to the network broadcast address are transmitted to every host with the same network number as the broadcast address. Broadcast packets are routinely used by the network to share routing information, field ARP requests, and send status and informational messages.

There are two common conventions used for broadcast addresses. The old convention, which older versions of SunOS and Berkeley UNIX BSD use, implements a broadcast address as the network portion of the address followed by all zeros. Using this convention, the broadcast address for the network 10.44 is 10.44.0.0. The new convention, which MultiNet and most other TCP/IP implementations use, implements a broadcast address as the network portion of the address followed by binary ones in all host portions of the address. In this scheme, the broadcast address for network 10.44 is 10.44.255.255.

If the network contains subnets, the broadcast address is relative to the local subnet. For example, host 10.44.12.1 with a subnet mask of 255.255.255.0 has an IP broadcast address of 10.44.12.255.

## Host Names

Most sites assign host names to each system on the network because names are easier to remember than IP addresses. On a small, locally contained network, a host name may be only one word, such as WILLOW. However, on larger networks or on networks connected to the Internet, names are longer and denote a place in the organization and ultimately on the Internet. These longer, more detailed names are



called fully qualified host names or fully qualified domain names (FQDNs). An example is BIGBOOTE.EXAMPLE.COM, where BIGBOOTE is the individual host (or system) name, EXAMPLE identifies the organization to which it belongs, and COM indicates this organization is involved in commerce on the Internet.

## TCP/IP Operation

The following steps present a highly simplified view of the events that occur during successful network communication.

1. Using the appropriate application, such as electronic mail, a user initiates communication to another system, identifying the remote system by name, such as BIGBOOTE.EXAMPLE.COM.
2. The application asks for the IP address of the system identified as BIGBOOTE.EXAMPLE.COM.
3. Using either DNS or host tables, the IP address of BIGBOOTE.EXAMPLE.COM is determined.
4. A connection is established using a three-way handshake.
5. Application information is organized into packets for transmission across the network.
6. The MTU (Maximum Transmission Unit) of the physical network is determined; if necessary, the packets are fragmented before being sent to the network interface card for delivery.
7. The hardware address of the next host (or hop) in the route to the target host is determined.
8. Each host along the route receives the packets and forwards them to the next hop in the route.
9. Once the packets arrive at the destination, they are reassembled in the appropriate order and delivered to the appropriate application. Some protocols acknowledge receipt of the packets to the sending host.

## Basic TCP/IP Protocols

Networking protocols ensure reliable delivery of information from one host to another.

This section describes several of the more important TCP/IP protocols.

- IP (Internet Protocol)
- IPv6 (Internet Protocol V6)
- TCP (Transmission Control Protocol)
- UDP (User Datagram Protocol)
- SLIP (Serial Line Internet Protocol)
- PPP (Point-to-Point Protocol)

# IP (Internet Protocol)

IP (Internet Protocol) is the networking protocol used to deliver data packets from one computer to another. The computers may reside on different networks as long as information can travel between them.

The IP layer in a TCP/IP stack is responsible for:

<p>Routing data packets from one system to the next until they reach their destination</p>	<p>When a packet is received, the IP layer examines its routing and interface tables to see if the IP address of the destination host is one of its own IP addresses or a broadcast address. If the destination IP address is the same as the local IP address, IP passes the packet to the TCP or UDP layer.</p> <p>If the IP address does not belong to this host and is not a broadcast address, the IP layer determines the next hop in the route. If this host is configured as a router, it forwards the packet to the next hop. If this host is not configured as a router, it discards the packet.</p>
<p>Discovering the MTU</p>	<p>The MTU (Maximum Transmission Unit) is the size of the largest packet that can be sent along the physical network. The MTU depends on the type of physical network being used. For example, a typical MTU for Ethernet networks is 1500 bytes, while a typical MTU for FDDI is 4352 bytes.</p> <p>When the IP layer receives a packet to send, it determines which route it will use to forward the packet and obtains that route's MTU.</p>
<p>Fragmenting and reassembling packets</p>	<p>If a packet is larger than the MTU, the IP layer is responsible for breaking the packet into smaller pieces or fragments that travel along the network. A fragment can be further fragmented as required by the next segment of the network.</p> <p>All reassembly occurs at the destination, where the IP layer is responsible for putting all the fragments together in the right order before passing the packets on to the TCP or UDP layer.</p>

## IPv6 (Internet Protocol Version 6)

IPv6 is an advancement on IP (v4) which supports a larger address space and has various efficiency and security improvements. It has the same responsibilities that IPv4 does in the stack and can be used by the TCP or UDP layer.

## TCP (Transmission Control Protocol)

TCP (Transmission Control Protocol) provides a reliable mechanism for delivery of information to remote hosts.

On the sending host, the TCP layer of the TCP/IP stack is responsible for:

- Organizing the information sent by the application into segments (the amount of data that will fit into an IP datagram)
- Specifying the endpoints (ports) of the connection with the remote host
- Establishing a connection with the remote host
- Ensuring the remote host acknowledges packets that have been sent within a specified time

On the receiving host, the TCP layer of the TCP/IP stack is responsible for:

- Acknowledging received packets
- Organizing the packets into the correct sequence upon receipt from the sending host
- Forwarding the packets to the application using the specified port

TCP requires more overhead than UDP but provides reliable delivery of packets to the remote host.

## UDP (User Datagram Protocol)

Applications can also use UDP (User Datagram Protocol) to deliver information to a remote host.

The UDP layer of the TCP/IP stack is responsible for:

- Organizing the information to be sent into a packet called a datagram
- Using a port to identify the program on the remote host to which the datagram is to be sent
- Verifying that the datagram contains the correct IP source and target addresses

UDP does not verify the successful delivery of packets to the target host. As a result, UDP requires less overhead than TCP. To accommodate this lack of verification, applications that use UDP often provide their own mechanisms for ensuring messages reach the target host in the correct sequence when required.

## **SLIP (Serial Line Internet Protocol)**

SLIP (Serial Line Internet Protocol) allows the transmission of IP packets over serial lines. SLIP can be used over a direct connection between the serial ports of two systems, or over telephone lines with modems.

## **PPP (Point-to-Point Protocol)**

Like SLIP, PPP (Point-to-Point Protocol) allows the transmission of IP packets over serial lines. PPP is a more versatile protocol than SLIP, and contains functionality that SLIP does not, such as:

- The ability to share the serial line with other protocols
- Error detection
- Support for both synchronous and asynchronous communication
- Dynamic configuration
- Negotiation of parameter values
- Support for different user-authentication protocols

While PPP is a more versatile serial-line protocol than SLIP, it is not available with all TCP/IP implementations.

# **Dynamic Configuration Protocols**

To communicate with the rest of the network, a host must have an IP address. However, some systems do not have the hardware to permanently store an IP address. In addition, computers frequently share IP addresses when there are more computers than IP addresses, or when IP addresses are used only temporarily. For these situations, there are three dynamic configuration protocols: RARP, BOOTP, and DHCP.

## **RARP (Reverse Address Resolution Protocol)**

RARP (Reverse Address Resolution Protocol) sends IP addresses to workstations that broadcast RARP requests containing their hardware addresses. RARP supplies IP addresses only and is commonly used by diskless workstations to determine their Internet addresses.

## **BOOTP (Bootstrap Protocol)**

BOOTP (Bootstrap Protocol) lets a host receive an IPv4 address and other configuration information from a BOOTP server on the network. BOOTP often specifies a bootstrap file for a client system to

download, normally via TFTP (Trivial File Transfer Protocol). BOOTP lets systems that have no hard disk retrieve the information necessary to access their bootstrap file.

## DHCP (Dynamic Host Configuration Protocol)

DHCP (Dynamic Host Configuration Protocol) builds upon the BOOTP protocol by letting a system receive all of the information necessary to function as a host on the network directly from a DHCP server. Unlike BOOTP, which only provides for permanent IPv4 addresses, DHCP supports three different mechanisms for allocating IPv4 addresses:

Automatic	Hosts requesting an IPv4 address receive a permanent IPv4 address
Dynamic	Hosts requesting an IPv4 address receive a temporary IPv4 address
Manual	IPv4 addresses are manually configured and DHCP delivers the assigned addresses to requesting hosts

## Routing

Routing is the process of selecting the path that data packets take to reach their destination. Routing can be as simple as delivering packets to another host on the same network (*direct routing*), or it may involve forwarding packets to routers on the way to the destination network. This section explains the basics of IP routing.

IP routing determines how to forward packets to a destination host. When a packet is forwarded to a local host (that is, a host on the same network), routing is *direct*; if the packet must be forwarded through one or more routers to reach its destination, the route is *indirect*.

Routing information for indirect routes is stored in a table of IP and router address pairs. Information in the routing table can be specified in three ways:

Static routes	Static routes are used to specify routing information explicitly. They are usually easy to maintain, but they provide no mechanism to respond automatically to changing environments.
Default routes	Default routes are used when a host has no specific route for the destination host or network in its routing table. If data cannot be delivered directly (because the routing table

	has no entry for the destination host or network), the data is forwarded to the default router.
Dynamic routing	Dynamic routing can use a service such as GATED to exchange routing information between cooperating systems. The protocols used to exchange information are RIP (Routing Information Protocol), EGP (Exterior Gateway Protocol), HELLO (DCN Local Network Protocol), and BGP (Border Gateway Protocol).

The following sections describe routing tables and GATED in more detail.

## The Routing Table

The routing table stores information about the routes that hosts can use to reach other hosts on the network or Internet. The routing table entries can be configured statically by the system manager, or dynamically by a program such as GATED.

- Static entries are established by manually entering information. Once a static routing table is established, you must update the table as changes occur.
- Dynamic entries are generated from information provided by a routing protocol (such as RIP) which collects information from other routers to populate the table. Dynamic routing solutions automatically share information and update the table as routing information changes.

The routing table is designed to supply the next hop address (which is always local) for data bound for other networks. The routing table never contains information about routers beyond the local network segment, nor does it contain information about how to reach individual host addresses (although it can contain host-specific entries). Routers always forward data to networks until the destination network is the local network. When the data arrives at the destination network, it is forwarded directly to the appropriate host.

Host-specific routes are special routing table entries that specify which router to use when data is bound for a specific remote host. Host-specific routes are frequently used to test new routers or to implement network security procedures.

## Router Discovery

Router discovery is a method of finding a router when no default route entry exists in the routing table. When booting, a host using router discovery broadcasts a message asking for available routers. The available routers reply with a message indicating their address. The host adds the information to its routing table and sets the default route based on advertisements from routers on the local network automatically. Local routers must also support RDISC (Router Discovery protocol).

Under IPv6 Router Discovery is performed by the RTSOLD program after line initialization.

## **GATED**

GATED can both learn and advertise known routes, allowing for automatic handling of network configuration changes and automatic selection of the best available route. Other routers on the local network must also support at least one of the protocols used by GATED (EGP, BGP, RIP, and HELLO). GATED only supports IPv4 routing.

# **DNS (Domain Name System) and Host Tables**

DNS (Domain Name System) and host tables are two methods of mapping between host (computer) names and their IP addresses. When you specify a host by name, DNS or host tables are used to map the host name to its IP address. The host name can be local to your organization or anywhere in the world, if your site is connected to the Internet. DNS and host tables can also be used to map IP addresses to host names.

## **DNS (Domain Name System)**

TCP/IP applications use DNS to convert host names to IP addresses, and vice versa. This conversion is called resolving.

A DNS resolver sends requests to another computer, called a DNS server, to resolve names into IP addresses. The DNS resolver can also send requests to the DNS server to resolve IP addresses to names.

DNS servers store host name and IP address information. If your computer needs information that is not on one DNS server, the server automatically requests the information from other servers.

## **Domains**

In DNS terminology, a domain is a group of computers. The domain administrator determines which computers are in the domain. A domain name identifies a domain and consists of words separated by dots. An example of a domain name is EXAMPLE.COM.

The parts of a domain name are created by the domain administrator or may be special words used on the Internet. Domain names can pertain to a site, an organization, or to types of organizations.

When read right to left, the first word in the domain name is the top-level domain which identifies the function of an organization or specifies a country name code. In the name EXAMPLE.COM, .COM

indicates an organization engaged in commerce. The top-level domain can also indicate a country, such as .FR for France, or .IT for Italy. The name of the organization is to the left of the top-level domain, such as EXAMPLE. Any words to the left of the top-level domain are called subdomains. The left-most word in the domain name is the host name. For example, in BIGBOOTE.EXAMPLE.COM, BIGBOOTE is a host in the EXAMPLE.COM organization.

Domains and subdomains are organized in a hierarchical tree structure. Just as the root directory in VMS is expressed as an implicit 000000., the root directory in DNS is expressed as a dot (.). Domains are analogous to directories; subdomains are analogous to subdirectories within directories.

Top-level domains such as .ORG, .COM, and .EDU exist in the United States. Other countries group their domain names below their two-letter country code. Domains grouped under country codes include domains such as .CO for commercial and .AC for academic. In the United States, .US is occasionally used instead of another top-level domain name. Subdomains may provide additional geographic information, such as .PALO-ALTO.CA.US.

## DNS Server

A DNS server is any computer running DNS software that lets it communicate with other DNS servers and store address information for later retrieval. DNS servers are also called name servers. Name servers cache (store) domain name information in memory for faster retrieval. Your network administrator provides the IP address of the name server on your network. Hosts implementing DNS come in five varieties:

Root name server	A root name server provides information about the start or base of the domain name tree. A root name server delegates authority to other primary name servers for the top-level domains such as .COM, .EDU, .US, .IT, etc. A root name server usually also handles those domains just below the root.
Primary name server	A primary name server has authority over one or more domains or subdomains. A primary name server reads information about the domain over which it has authority from the zone file, a special file that describes information about the domain and the hosts in that domain.
Secondary name server	A secondary name server for a domain receives information updates from the primary name server for that domain at regular intervals, and stores this information on disk. A secondary server is also authoritative for the domain.



Caching-only name server	A caching-only name server is not authoritative for any domain. If a caching-only name server cannot resolve a request, it forwards the request to an authoritative name server for that domain and caches the results for future use.
Resolver	A resolver sends requests for resolution to a DNS server. Any name server that can handle the request returns the response.

## Host Tables

If DNS is not configured on your network, you can configure MultiNet host tables to resolve names and addresses. Like DNS, host tables also map between IP addresses and host names; unlike DNS, however, the information is stored locally on your computer and must be updated manually. Using host tables, you must ensure that every host name you specify while running TCP/IP applications is listed with its IP address. Whenever a change occurs on the network, such as when a new computer is added that you need to access, you must add the information to the host table. With the growth of the Internet, maintaining host tables for it has become practically impossible.

When you add or modify a host table entry, you specify the host name, the IP address, an optional description, and one or more optional, alternative names (aliases) for the host.

## Using DNS and Host Tables Together

If you are using DNS, you may also want to use host tables. This is useful for temporary situations, such as when a new computer is added to the network, but has not yet been added to DNS.

The advantage of using DNS and host tables together for name resolution is that your system can access other systems even if the DNS server is not running or if the network is down. If you maintain entries in the host table for your local network, you can continue communicating with local systems until the DNS server or network is restored.



**Warning!** It is crucial to keep your host table entries synchronized with the DNS information.

# ARP (Address Resolution Protocol)

Before hosts can communicate with each other, the sending host must discover the hardware address of the receiving host.

Hardware addresses are unique numbers (for example, 00:DD:A8:13:48:C5) assigned to network interface boards by their manufacturers or by network administrators.

ARP (Address Resolution Protocol) discovers the hardware address corresponding to a specific IP address and dynamically binds the hardware address to the IP address.

ARP is a low-level protocol that lets network administrators assign IP addresses to hosts on a network as they see fit. There is no need to match the addresses to those on the physical network because ARP handles this process dynamically.

An ARP mapping (also called a *translation*) provides the correct delivery address (that is, the hardware address) on the network for data destined for an IP address. ARP mappings are stored in a table in memory known as the *ARP cache*.

When data is to be delivered to a local IP address (an IP address on the same physical network), the TCP/IP stack broadcasts an ARP request to all hosts on the local network segment. The request message asks all hosts if the IP address belongs to them. If the IP address belongs to a host on the local network segment, that host adds its hardware address to the packet and returns it to the sender. All other hosts on the network discard the request. The ARP cache stores the address resolution information returned and makes it available each time network data is bound for that IP address.

Old mappings are deleted from the ARP cache automatically after a short period of time. Old mappings are also deleted automatically when they no longer work (that is, when new, correct mappings become available).

## Neighbor Discovery

Neighbor Discovery is the IPv6 mechanism for mapping an IPv6 address to a hardware interface address. It works in similar ways to ARP does for IPv4, though it has improvements to reduce the impact on nodes other than the one that the address is being resolved for. Neighbor Discovery is also used as part of the Duplicate Address Detection portion of the autoconfiguration of interfaces.

# SNMP (Simple Network Management Protocol)

SNMP (Simple Network Management Protocol) allows you to manage remote hosts on a network (for example, routers, hubs, and workstations). Both the network management host and the managed hosts (called agents) must follow the SNMP rules. Because SNMP is an open standard, you can mix and match network management hosts and agents from different vendors.

SNMP maintains information about your workstation in a management information base (MIB).

## SNMP Traps

One of the main uses of SNMP is to make it easy to keep track of important events that occur on the managed network. To help automate network management, SNMP agents automatically send messages called traps to the network management host when certain events occur. For example, your workstation sends a trap when you reboot it.

One important type of SNMP trap is the *authentication failure trap*. Because SNMP network management hosts have access to sensitive configuration settings for the hosts on a managed network, it is important for network administrators to guard against breaches in network security that involve illegitimate use of SNMP messages.

For this reason, every SNMP message must be authenticated by network management hosts and SNMP agents using passwords called *communities*. If your agent receives an SNMP message that contains an incorrect community name for the type of operation requested, your agent sends a message to a network management host. This message contains information about the request your agent received:

- What the message requested
- Why your agent would not fulfill the request

## SNMP Communities

An SNMP community is a type of password used by the SNMP network management host and SNMP agents to ensure that only known and trusted hosts can send SNMP messages to and receive SNMP messages from each other. Every SNMP message includes a community name, so every message can be validated. There are three types of community names:

Read	The network management host must use the correct read community name when asking your SNMP agent to send it information about your host.
------	--

Write	The network management host must use the correct write community name when asking your SNMP agent to change some characteristic about your configuration.
Trap	If certain events happen in your workstation (for example, when you reboot your host, or when a network management host sends an SNMP message that contains the wrong read or write community name), your SNMP agent sends a trap message to a network management host. If your trap message is to be handled, the trap community name you send must match the name known to the target network management host.

# 5. Devices, Protocols, and MultiNet Internals

This chapter lists the devices and protocols supported by MultiNet and explains how the MultiNet kernel interacts with the OpenVMS Operating System.

## Devices Supported by MultiNet

MultiNet supports a variety of network topologies. Supported network interfaces are:

- HP Ethernet controller (shared)
- HP FDDI controller (shared)
- IP-over-DECnet link
- Asynchronous PPP using any OpenVMS-supported terminal multiplexer
- SLIP (Serial Line IP) using any OpenVMS-supported terminal multiplexer
- Turbochannel and PCI Token-Ring interfaces for OpenVMS Alpha
- Six-To-Four IPv6 over IPv4 tunneling interface
- HP Token-Ring adaptors on OpenVMS Alpha

## Protocols Supported by MultiNet

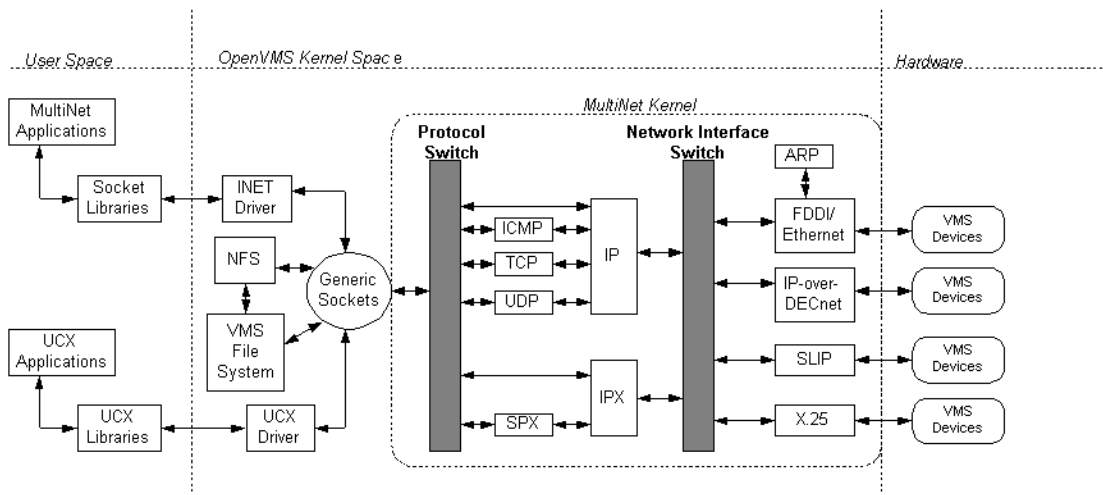
MultiNet is compatible with the current versions of the standard Internet networking protocol specifications listed below:

- BGP (Border Gateway Protocol): RFC-1105, RFC-1323
- BOOTP (Network Bootstrap Protocol): RFC-951, RFC-1534, RFC-1542, RFC-2132
- DHCP (Dynamic Host Configuration Protocol): RFC-2131, RFC-2132
- DNS (Domain Name Service): RFC-1034, RFC-1035, RFC-1101, RFC-1348
- EGP (Exterior Gateway Protocol): RFC-904
- Ethernet ARP (Address Resolution Protocol): RFC-826
- Ethernet RARP (Reverse Address Resolution Protocol): RFC-903

- FDDI (Fiber Distributed Data Interface): RFC-1188
- FINGER (User Status Protocol): RFC-1288
- FTP (File Transfer Protocol): RFC-959, RFC-1579, RFC-2389, RFC-2428
- ICMP (Internet Control Message Protocol): RFC-792, RFC-1256
- IP (Internet Protocol): RFC-791
- IP-over-X.25: RFC-87
- IPv6: RFC-2460, RFC-4294
- ICMP6. RFC 2463
- IPP (Internet Printing Protocol): RFC-2910, RFC 2911
- Kerberos: RFC-1411, RFC-1510
- NFS (Network File System): RFC-1094
- NTP (Network Time Protocol): RFC-1059
- OSPF (Open Shortest Path First): RFC-1583, RFC-1584
- Path MTU (Maximum Transmission Unit) Discovery: RFC-1191
- POP2 (Post Office Protocol Version 2): RFC-937
- POP3 (Post Office Protocol Version 3): RFC-1725
- PPP (Point-to-Point Protocol): RFC-1332, RFC-1552, RFC-1661
- RIP (Routing Information Protocol): RFC-1058
- RPC (Remote Procedure Call Protocol): RFC-1057
- SLIP (Serial Line Internet Protocol): RFC-1055, RFC-1144
- SMTP (Simple Mail Transfer Protocol): RFC-821, RFC-822, RFC-974
- SNMP (Simple Network Management Protocol): RFC-1157, RFC-1213
- SNMP Agent eXtensibility (RFC-2741, RFC-2742)
- SYSTAT (Active Users Service Protocol): RFC-866
- TCP (Transmission Control Protocol): RFC-793
- TELNET (network virtual terminal protocol): RFC-854, RFC-855, RFC-856, RFC-857, RFC-2941
- RFC-858, RFC-859, RFC-860, RFC-885, RFC-1041, RFC-1073, RFC-1079, RFC-1091, RFC-1096, RFC-1205, RFC-1372, RFC-1411, RFC-1416
- TFTP (Trivial File Transfer Protocol): RFC-1350
- TN3270: RFC-1576
- UDP (User Datagram Protocol): RFC-768
- WHOIS (Directory Service Protocol): RFC-954
- XDR (eXternal Data Representation): RFC-1014

# Understanding MultiNet Internals

This section describes how the MultiNet kernel interacts with the OpenVMS operating system. To understand the information in this section, some background in OpenVMS internals is helpful. The figure below illustrates the MultiNet protocols and the overall organization of the MultiNet kernel.



MultiNet interacts directly with the OpenVMS operating system. Generic sockets pass requests to the protocol switch, that differentiates between requests for IP use and for other support facilities. Requests are then sent through the network interface switch, which identifies the device for which the request is bound. When a request is received from a device, the steps occur in reverse.

## The \$QIO Interface

The programs implementing the lower layers of MultiNet - data link, network, and transport layers, with the exception of shared OpenVMS device drivers - reside in the MultiNet kernel. The MultiNet kernel is loaded into the OpenVMS S0 space (where the OpenVMS kernel is also loaded). Pages for S0 space are allocated when the OpenVMS system boots.

**Note:** You must use the `SPTREQ SYSGEN` parameter on VAX systems to set aside the appropriate number of S0 pages.

The MultiNet kernel accommodates multiple \$QIO interfaces. Each \$QIO interface is implemented by a separate OpenVMS pseudo-device driver, allowing MultiNet to simultaneously support the \$QIO

interfaces of several popular networking implementations. This lets you use, without modification, applications designed for these other networking implementations. The default MultiNet \$QIO interface, implemented in `INETDRIVER.EXE`, is used by the MultiNet shareable socket library and all MultiNet applications.

All MultiNet \$QIO drivers communicate with the MultiNet kernel through a set of kernel transfer vectors that interface with the generic socket layer. The generic socket layer of the MultiNet kernel provides most of the facilities common to all network protocols (including reading and writing user-level data and synchronizing OpenVMS I/O request packets with network protocol events).

The generic socket layer uses the protocol switch for protocol-specific operations. The protocol-specific code (which may consist of several interconnected protocol modules) calls through the network interface switch to the appropriate network device driver to encapsulate and transmit packets.

The protocol-specific code can also receive timer interrupts through the protocol switch. Incoming packets are decapsulated by the network device drivers and passed to the protocol-specific code through the network interface switch.

## Network Interface Device Drivers

The MultiNet kernel includes code allowing it to handle I/O for most network interface devices itself, rather than using OpenVMS device drivers. However, for devices that MultiNet shares with other software (for example, an Ethernet interface), the kernel uses standard OpenVMS device drivers and either VCI (VMS Communication Interface), FFI (Fast Function Interface), or ALTSTART driver interfaces.

## Custom Applications

The include and library files (optionally installed with MultiNet) provide access to several programming interfaces for writing custom client and server applications. These interfaces include:

- A 4.3BSD-compatible shareable socket library
- A 4.4BSD-compatible shareable socket library
- An RPC (Remote Procedure Call) library based on the public domain RPC library
- The standard MultiNet \$QIO interface
- A \$QIO interface compatible with TCP/IP Services for OpenVMS

Consult the *Programmer's Reference* for additional information about the socket library and the MultiNet \$QIO programming interface. The Sun RPC library routines are documented in the Sun Microsystems guide, *Networking on the Sun Workstation*.



# 6. Configuration Overview

This chapter presents an overview of MultiNet configuration tasks and the configuration tools that MultiNet provides.

## Configuration Tasks

MultiNet configuration consists of the following tasks:

<p>Establishing Basic IP Connectivity</p>	<p>Configures your system so it can reach other systems with commands such as FTP (for transferring files) and TELNET (for connecting to remote systems), both of which are described in the <i>MultiNet User's Guide</i>.</p> <p>The configuration options offered during MultiNet installation (see the <i>Installation and Introduction</i>) allow you to establish basic IP connectivity over a single Ethernet interface. To establish basic IP connectivity over any of the other MultiNet-supported network interfaces, see Chapter 11.</p>
<p>Configuring services</p>	<p>Allows other hosts to access resources on your MultiNet system such as OpenVMS print queues, files, and fonts for X displays.</p> <p>By default, installing MultiNet configures several of the most often used services.</p> <p>To learn about MultiNet services in general, see Chapter 12. For details on configuring specific services such as GATED (for dynamic routing), DNS (for Domain Name System-based host name lookup service), and FTP (for file transfers), refer to the appropriate chapters in this guide.</p>

# Configuration Utilities Overview

When MultiNet starts, it obtains configuration parameter values from a set of configuration files all found in the `MULTINET:` directory. Once MultiNet is running, you can change the configuration by modifying the current configuration in memory using the `MULTINET SET` command or by defining MultiNet logicals.

**Note:** Modifying the current MultiNet configuration using the `MULTINET SET` command and MultiNet logicals does not affect the MultiNet configuration files. Consequently, your changes are lost the next time unless you also modify MultiNet configuration files.

## Starting MultiNet

To start MultiNet, issue the following commands at the DCL prompt:

```
$ REPLY/ENABLE=NETWORK/TEMP  
$ @MULTINET:START_MULTINET
```

You must add an invocation of `START_MULTINET.COM` to the OpenVMS `SYSTARTUP` procedure, so MultiNet starts automatically the next time your system reboots.

To make sure MultiNet is running, use the `MULTINET CHECK` command. If the message "all tests passed" appears, MultiNet is running. If error messages appear, note the source of the errors, and refer to the corresponding chapter in this guide for details on modifying the configuration.

MultiNet starts with accounting enabled by default. To disable accounting in the Master Server, define the logical name `MULTINET_SERVER_NOACNT` with value 1 in the `SYSTEM` table before the Master Server starts.

## Restarting MultiNet

There are instances when you have to restart MultiNet. You can do this without rebooting VMS by executing the `START_SERVER.COM` command procedure in the MultiNet directory. This doesn't reinitialize any interfaces, it most likely will not affect any running processes, it will not reload the kernel.

**Note:** FTP and many other servers (POP3, REXEC, TELNET, etc.) will be unavailable for a short period so there could be brief interruptions with services.

## Modifying Configuration Files

MultiNet configuration utilities, which you invoke with the `MULTINET CONFIGURE` command, provide a command-line interface for each MultiNet configuration function.

**Note:** In general, you should never directly edit the configuration files. In the case of a network interface configuration, however, you may need to manually create and edit a device-specific configuration (see Chapter 11).

## Using the Command-Line Interface Utilities

Modifying MultiNet configuration files with the `CONFIGURE` utility's command-line interface consists of the following steps.

### Step 1: Checking Logicals

Before running `MULTINET CONFIGURE`, make sure the logical names `MULTINET` and `MULTINET_ROOT` are defined. These logical names are defined during installation but can be undefined manually or by rebooting without starting MultiNet automatically. To verify these logical names are defined, issue this command:

```
$ SHOW LOGICAL MULTINET*
```

If the logical names are undefined, issue this command:

```
$ @SYS$SYSDEVICE: [MULTINET.hostname.MULTINET] START MULTINET LOGICALS
```

If the logical names are already defined, executing this command does not cause a problem.

## Step 2: Starting MULTINET CONFIGURE

To start the command-line configuration utilities, enter:

```
$ MULTINET CONFIGURE /qualifier
```

*/qualifier* specifies the MultiNet feature you want to configure.

This table lists the tools available for editing configuration files and the commands that invoke them:

Configuration Type	Configuration Utility	DCL Command
Network Interfaces	NET-CONFIG	MULTINET CONFIGURE /INTERFACE
Electronic Mail	MAIL-CONFIG	MULTINET CONFIGURE /MAIL
Remote Print Queues	PRINTER-CONFIG	MULTINET CONFIGURE /PRINTERS
MultiNet Servers	SERVER-CONFIG	MULTINET CONFIGURE /SERVERS
MultiNet DECnet-over-IP Circuits	DECNET-CONFIG	MULTINET CONFIGURE /DECNET
MultiNet NFS Server	NFS-CONFIG	MULTINET CONFIGURE /NFS

Once the utility is started, its name appears as a prompt (such as NET-CONFIG>) at which you can enter commands.

## Step 3: Loading Configuration Files

By default, each command-line configuration utility modifies the configuration files that MultiNet reads when it starts. Usually, you need only modify configuration information and exit from the utility to change a configuration file.

To maintain multiple configuration files, however, use the configuration utility GET command to load and replace the configuration currently loaded. This feature is useful for abandoning modified configurations without quitting the utility.

For example, to load a test configuration file named MULTINET:TEST.CONFIGURATION into NET-CONFIG, enter:

```
NET-CONFIG>GET MULTINET:TEST.CONFIGURATION
```

After you modify the configuration, you can save the file under the same name or a different name.

**Note:** Although all command-line configuration utilities allow you to maintain multiple configuration files under different names, MultiNet only reads the standard configuration files, all of which are in the `MULTINET :` directory.

Feature	File and Description
BOOTP	BOOTP-SERVER.CONFIGURATION - Provides booting information for diskless hosts using the BOOTP protocol. This file can be edited; use a text editor to maintain this file.
Configuration	NETWORK_DEVICES.CONFIGURATION - Lists network devices and their configurations. This file is maintained by NET-CONFIG, and is read when MultiNet starts. <i>Do not edit this file!</i>
DHCP	DHCPD.CONF - Provides booting information for diskless hosts using the DHCP or BOOTP protocol. This file can be edited; use a text editor to maintain this file.
DNS	NAMED.CONF - Contains configuration information for the Internet DNS (Domain Name System) and the locations of other name service database files. This file can be edited; use a text editor to maintain this file.
General	MULTINET_VERSION - Contains the MultiNet version and revision level. <i>Do not edit this file!</i>
Host Tables	<p>HOSTS.LOCAL - Contains the local host table, with protocols and services for your local network. This file can be edited; use a text editor to maintain this file.</p> <p>HOSTS.SERVICES - Contains the official protocols and services supported by the local host (refer to RFC-943 for additional information). <i>Do not edit this file!</i></p>

	<p>HOSTS . TXT - Contains the DDN NIC host table, which lists the host names and Internet addresses of the Internet hosts known to the NIC. Because of the large size of this file, the HOSTS . TXT file provided with the MultiNet distribution includes an entry for NIC.DDN.MIL (the NIC information server system) only, rather than the entire file. You can retrieve a current copy of the NIC host table from NIC.DDN.MIL (formerly SRI-NIC.ARPA) via anonymous FTP. <i>Do not edit this file!</i></p> <p>The HOSTS . TXT file located on NIC.DDN.MIL is no longer maintained by the DDN NIC. <i>This file contains out-of-date information, and should be used with caution.</i> If your host is connected to the Internet, it is highly recommended that you use DNS to provide host-to-IP address mappings.</p> <p>HOSTTBLUK . DAT - Contains a compiled, binary version of the two host table files MULTINET : HOSTS . LOCAL and MULTINET : HOSTS . TXT. It contains the names of all hosts (in alphabetical order) in the host table files. MultiNet utilities refer to HOSTTBLUK . DAT when attempting to complete a partially specified host name. <i>Do not edit this file!</i></p> <p>NETWORK _ DATABASE - A compiled, binary version of the host, protocol, and services information contained in the files MULTINET : HOSTS . LOCAL, MULTINET : HOSTS . SERVICES, and MULTINET : HOSTS . TXT. The NETWORK _ DATABASE file allows applications to quickly look up the name or address of any host known in the host table files. <i>Do not edit this file!</i></p> <p>When DNS is disabled, the NETWORK _ DATABASE file provides the sole host lookup facility. When DNS is enabled, this database is only referred to if a name service query fails.</p>
NFS	<p>NFS . CONFIGURATION - Contains configuration information for the MultiNet NFS Client and NFS Server products. <i>Do not edit this file!</i></p>
NTP	<p>NTP . CONF - Contains configuration information for the Network Time Protocol (NTP) service. This file can be edited; use a text editor to maintain this file.</p>

Printing	REMOTE-PRINTER-QUEUES.COM - Configuration file for customizing printer queues. You must use MULTINET CONFIGURE/PRINTER to modify this file. You cannot edit it directly.
RARP	RARP.CONFIGURATION - Used by the MultiNet RARP server to determine Ethernet-to-IP address mappings so hosts and PCs can query the server to obtain their IP address. This file can be edited; use a text editor to maintain this file.
R services	HOSTS.EQUIV - Used by R services to perform authentication on incoming connections. This file can be edited; use a text editor to maintain this file.
Routing	GATED.CONF - Used by the Gateway Daemon (GATED) service to configure the RIP, EGP, HELLO, OSPF, and BGP dynamic IP routing protocols. This file can be edited; use a text editor to maintain this file.
Servers	SERVICES.MASTER_SERVER - Contains a list of the MultiNet servers and their configurations. This file is maintained by SERVER-CONFIG and is used by the MultiNet master server process. <i>Do not edit this file!</i>
SNMP	SNMPD.CONF - Contains configuration information for the MultiNet SNMP (Single Network Management Protocol) agent. Edit this file manually.
SMTP	SMTP_ALIASES - Contains SMTP (Simple Mail Transfer Protocol) mail alias and mailing list expansions. This file can be edited; use a text editor to maintain this file.  START_SMTP.COM and START_SMTP_LOCAL.COM - Contain SMTP configuration information. <i>Do not edit these files.</i>
SSH	SSH2_DIR:SSHD_CONFIG - Contains configuration information for the SSH server. This file can be edited; use a text editor to maintain this file.  SSH2_DIR:SSHD2_CONFIG - Contains configuration information for the SSH2 server. This file can be edited; use a text editor to maintain this file.

TELNET	MAP3270.DAT - Contains a keymap used by the TN3270 module of the TELNET client. The keymap maps keystrokes on standard ASCII terminals to IBM 327x special function keys. This file can be edited; use a text editor to maintain this file.
--------	---

## Step 4: Modifying the Configuration

Once you have loaded a configuration file into the configuration utility, you can modify the configuration with any commands the utility offers. Most utilities provide an assortment of SET and ADD commands. For guidelines on configuring with each utility, refer to the corresponding chapter in this guide.

For complete details on the commands available in each utility, refer to the corresponding chapter in the *MultiNet Administrator's Reference* or refer to the equivalent online reference by entering:

```
$ HELP MULTINET CONFIGURE qualifier
```

*qualifier* is the qualifier corresponding to the utility.

All of the TN3270 and TN5250 clients have been modified to properly handle large key mapping files like MAP3270.DAT and MAP5250.DAT without causing any access violations.

The extended TN3270 client has been modified to allow you to change its notion of the local language.

To use the extended TN3270 client, do the following:

```
$ DEFINE MULTINET TN3270 EMULATOR DPC EXTENDED
```

To change the local language,

```
$ DEFINE MULTINET DPC TN3270 LANGUAGE language
```

The *language* parameter can be one of the following:

BRAZILIAN	FRENCH CANADIAN	NEW HEBREW	SPANISH
BULTIN HEBREW	GERMAN	OLD BELGIAN	SPANISH SPEAKING
DANISH	ICELANDIC	OLD HEBREW	SWISS
FINNISH	ITALIAN	OLD PORTUGUESE	UK ENGLISH
FRENCH	NEW BELGIAN	PORTUGUESE	US ENGLISH



## Step 5: Verifying the Configuration

Because modified configuration files are not read until the next time MultiNet starts, it is useful to verify the validity of your changes before restarting. The NET-CONFIG and SNMP-CONFIG configuration utilities provide a CHECK command specifically for verifying your configuration. To verify changes made with either of these configuration utilities, use the CHECK command at the configuration utility prompt.

**Note:** All MultiNet configuration utilities automatically execute the CHECK function when you exit.

If there is a problem in the configuration, CHECK issues an error or warning message describing the problem.

- An ERROR message indicates a potentially fatal problem; for example, a problem that could cause network access to fail completely.
- A WARNING message indicates a less serious problem; for example, a problem that could cause the network to function in an unexpected manner.

If the utility displays error messages, use the SHOW command to view your changes and spot the error that caused the message. Correct the error using the configuration utility, then use the CHECK command to confirm the validity of the configuration.

The following example shows the automatic operation of CHECK after a NET-CONFIG session. Because the user executed an ADD command but specified a nonexistent OpenVMS device, CHECK issued an ERROR message. Because the user gave no IP address, CHECK issued a WARNING message.

```
NET-CONFIG>ADD SL1
[Adding new configuration entry for device "sl1"]
VMS Device [TTA0] TXA0
Baud Rate: [UNSPECIFIED] RETURN
Header Compression Mode: [DISABLED] RETURN
IP Address: [NONE] RETURN
Point-to-Point Device IP Destination Address: [NONE] RETURN
IP SubNet Mask: [NONE] RETURN
[add (Serial Line IP): Csr=NONE, Flags=%X0]
NET-CONFIG>EXIT
ERROR: sl1 can't $ASSIGN to SLIP Device:
%SYSTEM-W-NOSUCHDEV, no such device available
WARNING: sl1 has no protocol addresses specified
```

```
This network configuration FAILED the sanity check.  
Write startup file anyway ? [NO] NO  
$
```

## Step 6: Saving the Modified Configuration

After making changes to the configuration you have read in, use the configuration utility's `SAVE` or `WRITE` command to save the configuration either to the default or to non-standard files.

For example, to save network interface parameters, enter the following `NET-CONFIG` command:

```
NET-CONFIG>SAVE
```

**Note:** If you use the `EXIT` command to exit a configuration utility, the corresponding configuration files and startup command procedures are updated automatically.

If you do not want to save the edited configuration, you have two choices:

- Quit the utility without saving.
- Load the original configuration file back into the utility.

## Step 7: Exiting MULTINET CONFIGURE

To quit a configuration utility without saving your changes, use the `QUIT` command and enter `NO` when prompted to save the configuration.

To save the configuration and exit:

- Use the `EXIT` command, which automatically saves the configuration without prompting.
- Use the `QUIT` command and enter `YES` when prompted to save the configuration.

**Note:** Most configuration changes take effect the next time MultiNet starts. In some cases, however, such as when you add new network interfaces and change some global parameters, the changes take effect only after the system reboots. The configuration procedures in this guide tell you when to restart MultiNet or reboot your system.

## Putting Configuration File Changes into Effect

In almost all cases, you must restart some portion of MultiNet when you change MultiNet configuration files. In some cases, such as adding new network interfaces or changing some global parameters, you may have to reboot your system before the changes take effect.

Certain configurations of some features, such as the SNMP agent, can be reloaded into the current configuration without restarting the MultiNet master server process (`MULTINET_SERVER`).

The configuration procedures in this guide indicate when reloading, restarting the `MULTINET_SERVER` process, or rebooting your system is required.

## Modifying the Current Configuration

The `MULTINET SET` command provides a mechanism for knowledgeable system managers to manipulate the current configuration without restarting the `MULTINET_SERVER` process or rebooting the system. Use `MULTINET SET` with caution.

You can use `MULTINET SET` to:

- Modify local timezone information
- Modify DECnet-over-IP circuits
- Modify network interface configuration
- Manipulate the ARP table
- Manipulate the routing table

If you use the `/SNMP_HOST` qualifier, `MULTINET SET` can affect ARP tables, routing tables, and network interface configuration of remote hosts running a MIB-II-compliant SNMP agent, such as the MultiNet SNMP agent (see Chapter 23).

For a detailed description of `MULTINET SET`, refer to the *MultiNet Administrator's Reference*.

**Note:** Because `MULTINET SET` affects only the current configuration and does not affect the configuration files, any changes made with `MULTINET SET` are lost the next time you start MultiNet.



# 7. Host Tables and DNS

This chapter describes host tables and the Internet Domain Name System (DNS). Both host tables and DNS provide a way of associating host IP addresses with host names.

The chapter also describes how to use DNS load balancing for TCP-based services on cluster nodes. Load balancing helps to provide uninterrupted services if an individual server crashes or cannot handle the number of users trying to access it simultaneously.

## Methods of Associating IP Addresses and Host Names

The three methods of associating IP addresses and host names are:

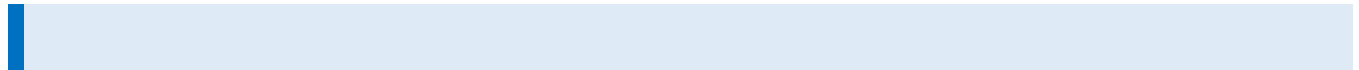
- Host tables, which offer a simplified method of translating a host name into an Internet address, can become unmanageable if there are many hosts on your network.
- DNS, which is more complicated to configure, offers the advantages of a distributed database.
- Multicast name resolution, which requires little configuration and is designed for small networks operating on a single logical LAN.

If you are connected to the Internet, DNS gives you access to the Internet DNS.

When you install MultiNet, you are asked if you want to use host tables or DNS. You can change your decision after installation using the instructions in this chapter.

DNS and the host table service are completely separate entities. If DNS is enabled, MultiNet only accesses host tables if a DNS query fails. DNS, however, never contacts the host table service, and no data is shared between these two services.

In addition to the administrative commands provided for configuring host tables and DNS, MultiNet provides C language library routines that can access either host tables or DNS. (See the *MultiNet Programmer's Reference* for more details.)



**Note:** For illustrative purposes ONLY, the section on host tables in this chapter assumes that you are not connected to the Internet, and the section on DNS assumes that you are connected to the Internet.

# Host Table

A host table is a file that describes network protocols, services, and host information accessed by utilities such as TELNET, RLOGIN, and TCPDUMP. To create a host table:

1. Build or retrieve the host table in one of the following ways:
  - Add site-specific information to the `HOSTS.LOCAL` file to describe protocol, service, network, and host characteristics. This method is described in the *Configuring the Host Table* section.
  - If you have a UNIX-style `/etc/hosts` file, convert it as described in the *Converting a UNIX /etc/hosts File* section.
2. Compile the host table file as described in the *Compiling the Host Table* section. The next time MultiNet starts, it installs the compiled host table as a global section.
3. To use the new, compiled host table, install it as a global section (see the *Installing a Compiled Host Table as a Global Section* section).

Once you install the compiled host table, MultiNet uses it to convert host names to IP addresses and vice versa. MultiNet's resolution of host names and addresses is invisible to users.

## Creating the Host Table Source Files

The first step in creating a host table is to gather the appropriate source files. There are three ways to obtain source files:

- Modify existing source files.
- Convert source files from other systems.
- Write completely new files.

# Host Table Source Files

The compiled form of the MultiNet host table is stored in the binary file `MULTINET:NETWORK_DATABASE`. The following list describes the source files from which the host table is compiled.

File name	Description
<code>MULTINET:HOSTS.LOCAL</code>	Contains any locally defined hosts, protocols, and services. Add custom host table entries to this file. For details on the source file format, see <i>Configuring the Host Table</i> .
<code>MULTINET:HOSTS.SERVICES</code>	Contains standard information necessary for the operation of MultiNet and is subject to change when you upgrade MultiNet. <i>Do not modify this file.</i>

All host table source files contain text in the form described by RFC-952, "DoD Internet Host Table Specification," with extensions provided to allow information not anticipated by the designers of RFC-952. For details, see the *Configuring the Host Table* section.

## Configuring the Host Table

The MultiNet host table contains more than just host name and address information. The below table shows the information stored in the MultiNet host tables and how to access the information.

The protocol and service definitions in the `HOSTS.LOCAL` file supplied with MultiNet are usually adequate. However, if you have an application that requires additional protocol or service information, you can add definitions for them. See *Adding Protocol Sections* and *Adding Service Definitions* later in this section.

You must add information about your network and hosts to the host table. See *Adding Network Definitions* and *Adding Host Definitions* later in this section.

The information stored in MultiNet host tables is:

Information	Keys	C Access Routine
IP Protocol Types	Name (for example, TCP, UDP)	<code>getprotobyname()</code> <code>getprotobynumber()</code>

	IP Protocol Number	
Services	Name (for example, TCP/TELNET) Port Number	getservbyname() getservbyport()
Networks	Name (for example, EXAMPLE-NET) IP Network Number	getnetbyname() getnetbyaddr()
Hosts	Name (for example, EXAMPLE.COM) IP Host Address	gethostbyname() gethostbyaddr()

## Adding Protocol Definitions

Protocol definitions in the MultiNet `HOSTS.LOCAL` file describe the protocols that can run on top of IP. A protocol definition contains the reserved word `PROTOCOL`, a protocol number used in the IP protocol field (for example, 6 for TCP), and the name of the protocol (for example, TCP). Enter each protocol definition as a single line without carriage returns or continuation characters.

The valid numbers and the protocol name values are defined in RFC-1060, "Assigned Numbers." Protocol numbers are 8-bit values ranging from 0 to 255. The protocol name can be up to 40 characters in length, and consists of uppercase letters, digits, and, optionally, a hyphen. Spaces and other special characters are not permitted in the protocol name. The format of a protocol definition is:

```
PROTOCOL : number : name :
```

For example:

```
PROTOCOL : 6 : TCP :
PROTOCOL : 17 : UDP :
```

## Adding Service Definitions

Service definitions in the `HOSTS.LOCAL` file describe the various protocol services that may be invoked, and the protocol and service port to contact for these services. A service definition consists of the reserved word `SERVICE`, the protocol name (for example, TCP), the port number (for example, 23 for TCP/TELNET), and the service name (for example, TELNET).



When specifying the service name, you may use a comma-separated list of names. The first name in the list is the official name of the service; the other names are aliases. Enter each service definition as a single line without carriage returns or continuation characters. Valid port number and service names are defined in RFC-1060, "Assigned Numbers." The format of a service definition is:

```
SERVICE : protocol : port : names :
```

For example:

```
SERVICE : TCP : 23 : TELNET :  
SERVICE : TCP : 25 : SMTP,MAIL :
```

## Adding Network Definitions

Network definitions in the `HOSTS.LOCAL` file correlate network names to network numbers. A network definition consists of the reserved word `NET`, the network number, and the network name. Network numbers are host IP addresses with the host part of the address set to zero. Enter each network definition as a single line without carriage returns or continuation characters. The format of the network definition is:

```
NET : network-number : name :
```

For example:

```
NET : 0.0.0.0 : DEFAULT-GATEWAY :  
NET : 192.16.100.0 : LOOPBACK-NET :  
NET : 192.41.228.0 : ABC-NET :
```

## Adding Host Definitions

Host definitions in the `HOSTS.LOCAL` file map IP addresses to host names. A host definition consists of the following:

- The reserved word `HOST`
- A comma-separated list of IP addresses by which the host may be contacted
- A comma-separated list of host names. The first host name is the official host name; any other names are aliases.
- The computer CPU type (used only as an informative message for users)
- The operating system (used only as an informative message for users)
- A comma-separated list of services provided by the host (currently ignored by all programs)

Enter each host definition as a single line without carriage returns or continuation characters. The format of a host definition is:

```
HOST : addresses : host names : CPU type : operating system : services :
```

For example:

```
HOST : 10.0.0.1 : EXAMPLE.COM,BIGBOOTE : VAX-11/780 : VMS : TCP/TELNET,  
TCP/FTP,TCP/SMTP :
```

Do not embed spaces in the CPU type, operating system, and offered-services fields. For example, "IBM-PC" is a valid CPU type and "IBM PC" is not a valid CPU type. Spaces are permitted before or after the colon separator character, but not within the field value. You may also leave these fields blank, as in the following entry:

```
HOST : 192.116.0.1 : LOCALHOST : : : :
```

## Host Name Conformance

Host names (A records) are now restricted to the following characters only:

Restricted	Excluded
A through Z (uppercase letters A through Z)	_ (underscore)
a through z (lowercase letters a through z)	/ (slash)
0 through 9 (the numbers zero through nine)	
. (dot or period)	
- (hyphen or dash)	

If there are any records in a zone file that do not meet these new guidelines, attempts for name resolution in that zone will fail. Other zones may begin to fail resolving your host names if your zone files are not in compliance with the relevant RFCs. RFC-952 (DoD Internet Host Table Specification, October 1985) and RFC-1123 (Requirements for Internet Hosts - Application and Support, October 1989) contain full descriptions of permitted host name specifications.

As a security measure, the current releases of the BIND nameserver enforce RFC-952 host name conformance (as modified by RFC-1123). As a result of this change, those host names that do not conform to the new rules will be unreachable from sites running the new nameserver. MultiNet properly responds to IGMP request messages because it is compliant with RFC 1112 "Host Extensions for IP Multicasting."

# Converting a UNIX /etc/hosts File

MultiNet provides a utility for converting a UNIX-format `/etc/hosts` host table file into a `HOSTS.TXT` file. Use the command procedure `MULTINET:CONVERT_UNIX_HOST_TABLE.COM` to make the conversion. This command has the following syntax:

```
$ @MULTINET:CONVERT_UNIX_HOST_TABLE infile outfile
```

- `infile` defaults to `HOSTS`.
- `outfile` defaults to the `HOSTS.TXT` file.

The `outfile` is converted to a file compliant with RFC-952, the DoD Internet Host Table Specification used by MultiNet. If `outfile` already exists, the command procedure replaces it with a new host table source file. During the conversion:

- Any comments in the input file that begin in the first column are preserved in the output file.
- Extra spaces are compressed.
- Blank lines are removed.
- Characters are converted to uppercase.
- If the host address or name cannot be found, an error is displayed and the output file is deleted.
- Host definitions are created with the host address, official name, and any aliases.

When the conversion completes, you may want to add the CPU type and operating system to each host entry, as described in *Adding Host Definitions* earlier in this section.

# Compiling the Host Table

After generating, modifying, or retrieving a MultiNet host table, compile it into binary form with the following command:

```
$ MULTINET HOST_TABLE_COMPILE
```

The command shown, the simplest form of the MultiNet host table compilation command, should suffice for most compilations. Additional qualifiers to this command are described in the `HOST_TABLE_COMPILE` command page listed in the *MultiNet Administrator's Reference*. The command qualifiers are:

Qualifier	Purpose	Default
<code>/HOST_TABLE_FILE</code>	Binary output file name	<code>MULTINET:NETWORK_DATABASE</code>
<code>/SILENTLY</code>	Suppress compilation messages	<code>NOSILENTLY</code>

/STARTING_HASH_VALUE	Initial hash size	Best value <sup>1</sup>
/TBLUK_FILE	Host-completion database	MULTINET:HOSTTBLUK.DAT
/UNIX_HOST_FILE	Produce UNIX-style hosts file	NUNIX_HOST_FILE

**Note:** If you are running MultiNet on a VMScluster, you only need to run the `MULTINET COMPILE` command on one node of the cluster.

## Installing a Compiled Host Table as a Global Section

When MultiNet starts, it installs the compiled host table `MULTINET:NETWORK_DATABASE` as a global section. The compiled host table is organized as a "perfect hash" lookup system, allowing MultiNet to answer any query in one lookup. Because the host table is installed as an OpenVMS global section, access to host table information is extremely fast. To install the compiled host table as a global section without restarting MultiNet:

1. After recompiling a host table, reinstall it by rebooting or invoking `@MULTINET:INSTALL_DATABASES`.
2. If you want the new host table to be noticed by the servers that run as part of the `MULTINET_SERVER` process, restart that process with `@MULTINET:START_SERVER`.
3. If you want the new host table to be noticed by the SMTP symbiont(s), restart them with the command `@MULTINET:START_SMTP`. For more information about configuring SMTP queues, see Chapter 15.

<sup>1</sup> The "best value" default is computed from the size of the data as the utility attempts to create 512-byte units. When you run `HOST_TABLE COMPILE`, the hash value is listed in the displayed messages. If you only added hosts and want to select a number for this qualifier, use the value from the previous compilation as a starting point.

**Note:** You must run the @MULTINET:INSTALL\_DATABASES, @MULTINET:START\_SERVER, and @MULTINET:START\_SMTP commands on every VMSccluster node running MultiNet.

# Using the Domain Name System (DNS)

DNS (Domain Name System) is the preferred method of maintaining host name and address information. DNS provides a fully distributed database of host names, Internet addresses, host information, and mail forwarding information.

When using DNS, a host has access to the full database, yet local information can be maintained locally and exported to the rest of the Domain Name System. DNS is fully documented in several RFCs published by the Defense Data Network Network Information Center (DDN NIC). The following RFCs describe DNS:

RFC-1032	Domain Administrators Operations Guide
RFC-1033	Domain Administrators Guide
RFC-1034	Domain Names - Concepts and Facilities
RFC-1035	Domain Names - Implementation and Specification

The Defense Data Network Information Center (DDN NIC) publishes a softbound manual containing all RFCs pertaining to DNS. This chapter cannot describe all of the intricacies of the Domain Name System. For complete information, Process Software recommends the book *DNS and BIND, Fourth Edition* (ISBN 0-596-00158-4). This book provides a thorough description of DNS concepts and applications, although the emphasis is on UNIX-based DNS implementations. In particular, you should read this book if you are using MultiNet to set up an authoritative, or master, name server for your domain or part of your domain.

# DNS Resolvers and Servers

MultiNet provides both DNS server and resolver (client) support. The server communicates with the rest of the Internet DNS and participates in distributing DNS data. The resolver receives requests from applications and queries a DNS server for the information.

## The DNS Resolver (Client)

The DNS resolver (client) is used by applications to access the DNS database. The DNS resolver is accessed via the `MULTINET_SOCKET_LIBRARY`.

The MultiNet DNS resolver is enabled when you use the `NET-CONFIG SET DOMAIN-NAMESERVERS` command. This command defines the domain name servers that your system queries to satisfy host-name-to-address translation, and is normally set to the local host. MultiNet is initially configured to provide a caching-only name server, which works with the resolver. When the resolver fetches a mapping, it is "cached" by the name server and stored for other applications that need the information.

The resolver is almost always invisible to an applications programmer. When using the MultiNet socket library, the normal host table access routines, such as `gethostbyname()` and `gethostbyaddr()`, automatically call the DNS resolver routines and only use host table access when a DNS resolver fails. For the MultiNet OpenVMS/ULTRIX Connection (UCX) \$QIO Driver, the \$QIO functions for translating host names and addresses are referred to as the MultiNet `MULTINET_SERVER` process, which then uses the DNS resolver routines to satisfy the UCX name translation query.

The MultiNet SMTP symbiont queries the Internet DNS for mail forwarding information and calls the DNS resolver routines directly to query the DNS for Mail Exchanger (MX) resource records.

## DNS Server

The MultiNet DNS server is based on the ISC BIND 9 Nameserver. The `MULTINET_SERVER` process starts the `NAMED_SERVER` process if the nameserver process is not running. The DNS server processes queries from resolvers, then responds to the queries, or queries other DNS servers to obtain information from other parts of the DNS database.

MultiNet is shipped with DNS configured to operate as a caching-only server in the Internet environment.

The MultiNet DNS server uses the `MULTINET:NAMED.CONF` file for configuration information. The configuration file is the equivalent of the `/etc/named.conf` file used by UNIX-based BIND implementations.

The configuration file typically contains references to other files that contain definitions for the server's contribution to the DNS database. These other files contain text in standard "resource record" format, as described in RFC-1035, "Domain Names - Implementation and Specification."

The ISC BIND 9 distributed HTML documents are on the CD-ROM in the directory named [BIND9-DOC]. Refer to the file named INDEX.HTML for a list of BIND documents.

## Enabling a Caching-Only Name Server

A caching-only name server queries other nameservers to resolve host names to IP addresses. The answers received from the inquiry are retained and used in subsequent name resolver requests without querying the remote name server. The default DNS configuration files are shipped as a caching-only server; you only need to enable DNS.

**Note:** Master, or authoritative, name servers also cache responses to their queries.

Use a caching-only name server if your OpenVMS system is not the authoritative name server for any domain.

To determine whether DNS is enabled on your system, check the MULTINET\_NAMESERVERS logical name:

```
$ SHOW LOGICAL MULTINET_NAMESERVERS
```

If DNS is enabled, your system is already set up to use DNS to resolve host names and addresses. If the logical is set to 127.0.0.1, your system also acts as its own name server.

If DNS is not enabled, you can enable it to take effect when the system is rebooted or to take effect immediately.

To enable the caching-only name server to take effect when the system reboots:

```
$ MULTINET CONFIGURE
```

```
MultiNet Network Configuration Utility 5.6(nnn)  
[Reading in MAXIMUM configuration from MULTINET:MULTINET.EXE]
```

```
[Reading in configuration from MULTINET:NETWORK_DEVICES.CONFIGURATION]
NET-CONFIG>SET DOMAIN-NAMESERVERS 127.0.0.1
NET-CONFIG>SET HOST-NAME fully-qualified-domain-name
```

Also ensure the fully qualified domain name (FQDN) is included in your host table. The easiest way to do this is to modify the HOST line that describes your system in MULTINET:HOSTS.LOCAL to be of this form:

```
HOST : address(s) : FQDN,short_name : [CPU] : [OS] : [services] :
```

See *Adding Host Definitions* earlier in this section for more information. If you change the MULTINET:HOSTS.LOCAL file to include your FQDN, you must also recompile the table and install it as a global section, as shown in this example:

```
$ MULTINET_HOST_TABLE_COMPILE
$ @MULTINET:INSTALL_DATABASES
```

To make the change take effect without rebooting:

1. Define the logical name MULTINET\_NAMESERVERS as 127.0.0.1 to take advantage of your local name server's cache:

```
$ DEFINE/SYSTEM/EXECUTIVE MULTINET_NAMESERVERS "127.0.0.1"
```

2. Define the official, fully qualified domain name:

```
$ DEFINE/SYSTEM/EXECUTIVE_MODE MULTINET_HOST_NAME "fqdn"
$ DEFINE/SYSTEM/EXECUTIVE_MODE ARPANET_HOST_NAME "fqdn"
$ DEFINE/SYSTEM/EXECUTIVE_MODE UCX$INET_HOST "name"
$ DEFINE/SYSTEM/EXECUTIVE_MODE TCPIP$INET_HOST "name"
$ DEFINE/SYSTEM/EXECUTIVE_MODE UCX$INET_DOMAIN "domain"
$ DEFINE/SYSTEM/EXECUTIVE_MODE TCPIP$INET_DOMAIN "domain"
```

3. Define the UCX equivalents:

```
$ DEFINE/SYSTEM/EXECUTIVE_MODE UCX$BIND_DOMAIN "domain-name"
$ DEFINE/SYSTEM/EXECUTIVE_MODE UCX$BIND_SERVER000 "127.0.0.1"
$ DEFINE/SYSTEM/EXECUTIVE_MODE TCPIP$BIND_DOMAIN "domain-name"
$ DEFINE/SYSTEM/EXECUTIVE_MODE TCPIP$BIND_SERVER000 "127.0.0.1"
```

4. Restart the MULTINET\_SERVER process:

```
$ @MULTINET:START_SERVER
```

5. Restart the SMTP symbiont(s):

```
$ @MULTINET:START SMTP
```



# Caching-Only Name Server Configuration with Forwarders

A caching-only name server usually sends queries directly to the name server that contains the answer. A `forwarders` option can be used to redirect these queries to a central name server that accepts recursive queries from other servers and functions as a second-level cache. The central name server then queries the name server that contains the answer, and caches a copy.

Configure caching-only name servers with forwarders in a network with multiple caching servers to:

- Reduce load on your connection to the Internet.
- Improve DNS response to repeated queries.

**Note:** Although adding `forwarders` statements improves DNS response times, DNS does not require it.

If the DNS server configuration file specifies one or more forwarders, the server sends all queries for data not in the cache to the forwarders.

Use a text editor to add a `forwarders` statement to the `options { };` section of the `NAMED.CONF` file to forward for all zones. Forwarding can also be specified on a per-zone basis, or turned off on a per-zone basis using `forwarders { };`.

For example, to add two servers with these IP addresses 192.1.1.98 and 192.1.1.99, the `forwarders` statement would be:

```
forwarders { 192.1.1.98; 192.1.1.99;};
```

There is also a `forward` option that tells the server how to use the forwarders. The `forward` option is only meaningful if the `forwarders` list is not empty. There are two values to use with the `forward` option.

Value	Description
<code>forward first;</code> (default)	The server queries the forwarders first. If the forwarders fail to find an answer, the server queries the root servers.

forward only;	The server queries the forwarders only. If the forwarders fail to find an answer, the server does not query the root domain servers.
---------------	--

For example, to use the forwarders only:

```
options {
    forward only;
    forwarders { 192.1.1.98; 192.1.1.99; };
};
```

After adding the forwarders and forward lines, restart the nameserver.

```
$ REPLY/ENABLE=NETWORK/TEMPORARY
$ MULTINET NETCONTROL DOMAIN RESTART
```

## Using a Search List to Resolve Host Names

When you specify a simple host name, the DNS resolver expands automatically the simple name by appending the local host's domain to it. For example, if your host is `bos1.example.com` and you enter this command: `$ TELNET sfo1` the DNS resolver expands `sfo1` automatically into `sfo1.example.com` and attempts to translate that name into an address.

To have a different domain appended to host names, or to search multiple domains, define the `MULTINET_SEARCHDOMAINS` logical. If you specify multiple domains, separate them with blanks. For example, to have the resolver search for simple names in the domains `sub1.example.com` and `sub2.example.com` instead of the local domain, use the following logical name definition:

```
$ DEFINE MULTINET_SEARCHDOMAINS "sub1.example.com sub2.example.com"
```

This logical can be defined by individual users or as a system-wide logical (with the `/SYSTEM` qualifier). The search list can be up to 511 bytes in length. To search the local domain in addition to other domains, include the local domain in the `MULTINET_SEARCHDOMAINS` list. The maximum number of domains to search is 6.

## Setting Up a Master Name Server

The following procedure sets up a master, or authoritative, name server. This type of name server gets data for its zone from files on the host where it runs. A zone is the domain, or portion of a domain, for which the master server has complete information. To set up a master server:

1. Determine whether DNS is enabled on your system by checking the logical name

`MULTINET_NAMESERVERS`:

```
$ SHOW LOGICAL MULTINET_NAMESERVERS
```

If `MULTINET_NAMESERVERS` is defined, your system uses DNS to resolve host names and addresses.

If `MULTINET_NAMESERVERS` is `127.0.0.1`, your system also acts as its own DNS server.

2. If it is not already enabled, enable DNS by specifying one or more DNS servers with `NET-CONFIG` using the `SET DOMAIN-NAMESERVERS` command.

**Note:** The name server list can include up to three IP addresses.

3. Make sure the official host name configured in `MULTINET_CONFIGURE` is the fully qualified domain name of the name server. If not, define it without rebooting using the following command:

```
$ DEFINE/SYSTEM/EXECUTIVE_MODE MULTINET_HOST_NAME "fully qualified domain name"
```

4. To specify the time between DNS name server requests to a nonresponding server and the number of attempts to make to re-establish communications, use the `NET-CONFIG SET NAMESERVER-RETRANSMISSION` command.

5. Update the DNS configuration file, `MULTINET:NAMED.CONF`, to add information about your site, as described in the `MULTINET:NAMED.CONF` section.

6. For each zone of type master, add a zone file to describe the zone characteristics. A zone is a range of authority that includes one or more fully qualified domains or part of a domain. A zone file describes the contents of a zone. Configuring a zone information file is described in the *DNS Zone Information Files* section.

7. Reload the DNS server (see the *Reloading the Name Server* section).

To prevent log files from being truncated when they are closed, use the logical `MULTINET_SERVER_DONT_TRUNCATE_LOG`. Note that the default allocation quantity for the log files is 80 blocks, so they could have a moderate amount of empty space.

## Domain Name versus Host Name

Your domain name should not also be the primary name of a host on your network. This is because of a possible conflict between the host name `LOCALHOST` and a registered Internet domain.

For example, if your domain name is `example.com`, and you want to have a host with the same name, give the host a different primary name, such as `main.example.com`. Create an appropriate A (address) and PTR (inverse lookup) entry with the name `main.example.com`. Then, add an A record for `example.com` that also points to the address for `main.example.com`.

If you already have a host with the same name as your domain name and the host is running MultiNet, configure a local domain for the MultiNet resolver on that system. For example:

```
$ MULTINET CONFIGURE
NET-CONFIG>SET LOCAL-DOMAIN example.com
NET-CONFIG>EXIT
```

## The MULTINET:NAMED.CONF File

The main DNS configuration file, from which the name server gets its initial data, is `MULTINET:NAMED.CONF`. The equivalent of this file in UNIX-based BIND implementations is `/etc/named.conf`. Use this file to add information about your site when setting up a master DNS server. An example configuration file follows.

```
/*
** Sample Configuration File for DNS server
*/

options {
    directory "SYS$SYSROOT:[MULTINET]";
    // forward only;
    forwarders { 128.0.1.1; 128.0.2.10; };
};

zone "example.com" in {
    type master;
    file "domain-name-service.bos";
};

zone "0.128.in-addr.arpa" in {
    type master;
    file "domain-name-service.bos-net";
};

zone "sfo.example.com" in {
    type slave;
    masters { 128.0.1.1; };
    file "domain-name-service.sfo";
};

zone "1.0.128.in-addr.arpa" in {
    type slave;
    masters { 128.0.1.1; };
};
```

```

    file "domain-name-service.cc-net";
};

zone "0.0.127.in-addr.arpa" in {
    type master;
    file "domain-name-service.local";
};

zone "localhost" in {
    type master;
    file "domain-name-service.localhost";
};

zone "." in {
    type hint;
    file "domain-name-service.cache";
};

```

The following sections describe the zone, options, and logging sections.

## Zone

A *zone* is that part of a name server that contains complete information about the domain name space. You specify a zone in the following way:

```

zone "domainname" [class] {
    type type;
    [other statements...]
};

```

NAMED.CONF zone fields defines the NAMED.CONF zone fields.

Field	Description
<i>class</i>	<p>The class to which this zone applies. If the class is not specified, the type IN is used by default. The syntax is [ ( in   hs   hesiod   chaos ) ]:</p> <ul style="list-style-type: none"> <li>in (default) - Used for objects connected to the Internet. This is the only supported type.</li> <li>hs or hesiod - Confined mostly to MIT. hs is the abbreviation for hesiod.</li> <li>chaos - An historic network. Not used today.</li> </ul>
"domain name"	The name of the domain for which this zone is authoritative.

### Zone statements:

Statement	Description
<code>file "filename";</code>	Specifies the name of the zone file.
<code>masters [ port ip_port ] { ip_addr; [ ip_addr; ...  ]};</code>	Specifies the IP address(es) and port from where the server is to transfer the zone data. This statement is meaningful only for slave or stub zones.
<code>type (master   slave   stub   hint   forward);</code>	See the Zone Types table for a description of these zones.

### Optional Zone Statements:

Statement	Description
<code>allow-query { address_match_list; }; allow-transfer { address_match_list; };</code>	Overrides the <code>allow-query</code> and the <code>allow-transfer</code> statements in the global options section for this zone. See <i>NAMED.CONF Options</i> .
<code>allow-update { address_match_list; };</code>	Specifies the addresses of hosts that are allowed to modify the zone with dynamic updates. Defaults to none.
<code>also-notify { ip_addr; [ ip_addr; ... ] };</code>	Overrides the “also-notify” statement in the global options section for this zone. See <i>NAMED.CONF Options</i> .
<code>check-names (warn   fail   ignore);</code>	Overrides the default name checking specified in the global options section. See the <code>check-names</code> statement in <i>NAMED.CONF Options</i> for more details.

<code>forward (only   first); forwarders {[ip_addr;...]};</code>	Overrides the <code>forward</code> and <code>forwarders</code> statements in the global options section for this zone. See <i>NAMED.CONF Options</i> .
<code>ixfr-base path_name;</code>	Specifies the file name used for the IXFR transaction log file.
<code>notify [(yes   no); ]</code>	Specifies if zone change notifications should be sent to the slave servers for the zone. This overrides the <code>notify</code> statement in the global options section. See the <code>notify</code> statement in <i>NAMED.CONF Options</i> for more details.
<code>pubkey flags protocol algorithm key;</code>	Represents a public key for this zone. The key is needed when this is the top level authoritative zone served by this server and there is no chain of trust to a trusted key. The key is considered secure, so data it signs will be considered secure. The DNSSEC flags (number), protocol (number), and algorithm (number) are specified, as well as a base-64 encoded string representing the key.
<code>transfer-source ip_addr;</code>	Overrides the <code>transfer-source</code> statement in the global options section for this zone. See <i>NAMED.CONF Options</i> .

### Zone Types:

Type	Description
<code>forward</code>	<p>A forward zone is use to direct all queries in it to other servers. The specification of the options in such a zone overrides any global options declared in the <code>options</code> statement.</p> <p>If a <code>forwarders</code> statement is not present in the zone or an empty list for <code>forwarders</code> is given, forwarding is not done for the zone, cancelling the effects of any <code>forwarders</code> in</p>

	<p>the <code>options</code> statement. Thus, if you want to use this type of zone to change the behavior of the global <code>forward</code> option, and not the servers used, you need to re-specify the global forwarders.</p>
hint	<p>The <code>MULTINET:NAMED.CONF</code> file example specifies that data in the <code>DOMAIN-NAME-SERVICE.CACHE</code> file, which is in standard resource record format, should be placed in the bootstrap cache. The hint zone definition is used to specify locations of root domain servers. An up-to-date list of root name servers is obtained automatically and stored in memory without replacing the cache file.</p>
master	<p>Specifies data for the zone and the domain. The first master zone definition in the example states that the file <code>DOMAIN-NAME-SERVICE.BOS</code> contains authoritative data for the <code>EXAMPLE.COM</code> zone, in standard resource record format.</p> <p>The second master zone definition in the example states that the file <code>DOMAIN-NAME-SERVICE.BOS-NET</code> contains authoritative data for the domain <code>0.128.IN-ADDR.ARPA</code>, which is used in translating addresses in network 128.0.0.0 to host names.</p> <p>Each zone master file should begin with an SOA (Start Of Authority) resource record for the zone, as shown in the <i>Error! Reference source not found.</i> section.</p>
slave	<p>Specifies the zones for which this DNS server acts as a secondary name server. After this name server receives a "zone transfer," it becomes authoritative for the specified zone.</p> <p>The first slave zone definition in the example on page 10-13 specifies that all authoritative data under <code>CC.FLOWERS.COM</code> is to be transferred from the name server at 128.0.1.1.</p> <p>The file statement in this section is the file name in which to back up the transferred zone. When it boots, the name server loads the zone from this backup file, if it exists, providing a complete copy even if the master DNS server is unreachable. This file is updated whenever a new copy of the domain is received by automatic zone transfer from one of the master servers. The file statement is optional, but recommended to speed up server startup and eliminates needless bandwidth.</p>



	The second slave zone definition in the example states that the address-to-hostname mapping for the subnet 128.0.1.0 should be obtained from the same list of master servers as the previous zone.
stub	Works like a <code>slave</code> zone, except it only transfers the nameserver records for the master zone rather than the full zone information.

## Options

The `options` statement sets up global options to be used by BIND. Use this statement only once in a configuration file. If it is used more than once, the first occurrence determines what options to use, and a warning is generated. If no `options` statement is present, an options block is used setting each option to its default value. You specify options in the following way:

```
options {
    options statements
};
```

**Note:** The following table shows some of the more commonly used option statements. For a full listing of the option commands that can be specified, please consult a detailed text such as O'Reilly's "DNS and BIND", or else consult the BIND RFC.

This table defines the `NAMED.CONF` options:

Option	Description
<code>allow-query</code> <code>{address_match_list;};</code>	See the <i>Address_match_list</i> section.  Specifies the addresses of hosts that are allowed to query the server for information. It defaults to all.

<pre>allow-recursion {address_match_list};;</pre>	<p>See the <i>Address_match_list</i> section.</p> <p>Specifies which hosts are allowed to make recursive queries through this server. If not specified, recursive queries from all hosts are allowed (default).</p>
<pre>allow-transfer {address_match_list};;</pre>	<p>See the <i>Address_match_list</i> section.</p> <p>Specifies the addresses of hosts that are allowed to perform zone transfers from the server. It defaults to all.</p>
<pre>also-notify {ip_addr; [ip_addr; ...]};</pre>	<p>Defines a global list of IP addresses that get sent NOTIFY messages whenever a fresh copy of the zone is loaded. This ensures that copies of the zones converge quickly on “stealth” servers. If an <code>also-notify</code> list is given in a zone statement, it overrides the <code>options also-notify</code> statement. When a zone <code>notify</code> statement is set to NO, the IP addresses in the global <code>also-notify</code> list are not sent NOTIFY messages for that zone. The default is the empty list (no global notification list).</p>
<pre>blackhole {address_match_list};;</pre>	<p>See the <i>Address_match_list</i> section.</p> <p>Specifies a list of addresses the server will not accept queries from or use to resolve a query. Queries from these addresses will not be responded to.</p>
<pre>check-names (master   slave   response) (warn   fail   ignore);</pre>	<p>The server checks names in three areas:</p> <ul style="list-style-type: none"> <li>• Master zone files.</li> <li>• Slave zone files.</li> <li>• Responds to queries the server has initiated.</li> </ul> <p>The server assumes the following defaults:</p>

	<pre>options {   check-names master fail;   check-names slave warn;   check-names response ignore; };</pre> <p>ignore - No checking is done.</p> <p>warn - Names are checked against their expected client contexts. Invalid names are logged, but processing continues normally.</p> <p>fail - Names are checked against their expected client contexts. Invalid names are logged, and the offending data is rejected.</p> <p>If <code>check-names response fail</code> has been specified, and answering the client's question requires sending an invalid name to the client, the server sends a REFUSED response code to the client instead.</p>
<pre>directory "path";</pre>	<p>Causes the server to change its default directory to the specified directory. This can be important for the correct processing of <code>\$INCLUDE</code> files in primary zone files, or file statements in zone definitions.</p>
<pre>fake-iquery ( yes   no );</pre>	<p>BIND 9 does not do IQUERY simulation. Obsolete.</p>
<pre>fetch-glue ( yes   no );</pre>	<p>Obsolete for BIND 9.</p>
<pre>forward ( only   first );</pre>	<p>This statement is meaningful only if there is a non-empty <code>forwarders</code> statement.</p>

	<p>When <code>first</code> (default) is used, the server queries the forwarders first before consulting the root domain servers.</p> <p>When <code>only</code> is used, the server queries the forwarders only. If the forwarders fail to find an answer, the server does not query the root domain servers. For example:</p> <pre>options {     forward only;     forwarders         { 192.1.1.98; 192.1.1.99; }; };</pre>
<pre>forwarders {ip_addr; [ip_addr; ... ]};</pre>	<p>Specifies the addresses of site-wide servers that accept recursive queries from other servers. If the DNS server configuration file specifies one or more <code>forwarders</code>, the server sends all queries for data not in the cache to the forwarders.</p> <p>Central name servers designated to handle forwarded requests can then develop a cache of answers to external queries. The central cache reduces the number of requests sent to root name servers and improves DNS performance.</p>
<pre>listen-on [port ip_port] {address_match_list; };</pre> <p>also see <i>listen-on-v6</i></p>	<p>See the <i>Address_match_list</i> section.</p> <p>Specifies what port on what interface to listen on. The default is:</p> <pre>listen-on port 53 { any };</pre> <p>For example:</p>

	<pre>options {     // listen on port 53 for     // external interfaces.     listen-on { 10.0.0.0; };     // listen on port 43 for     // internal interfaces.     listen-on port 43         { 127.0.0.1; 10.0.0.0; }; };</pre>
<pre>provide-ixfr (yes   no );</pre>	<p>If yes, a transaction log is kept for incremental zone transfer of each zone. The default is yes.</p>
<pre>max-ixfr-log-size number;</pre>	<p>Obsolete for BIND 9.</p>
<pre>max-transfer-time-in number</pre>	<p>Terminates the inbound zone transfers (named-xfer processes) running longer than the minutes specified. The default is 120 minutes (2 hours).</p>
<pre>min-roots number;</pre>	<p>Obsolete for BIND 9.</p>
<pre>notify ( yes   no );</pre>	<p>If yes (default), the server notifies slave servers if there are any changes to a domain for which the server is master or slave. The server determines the slave servers by the nameserver records contained in the zones data file.</p> <p>For more information, see the <i>also-notify</i> statement.</p>
<pre>query-source [address ( ip_addr   * )] [port ( ip_port   * )] ;</pre> <p>see also <i>query-source-v6</i></p>	<p>If the server does not know the answer to a question, it queries other nameservers. <code>query-source</code> specifies the address and port used for such queries. If address is * or is omitted, a wildcard IP address (INADDR_ANY) is used. If port is * or is omitted, a random unprivileged port is used. The default is</p> <pre>query-source address * port *;</pre>

	<p><b>Note:</b> <code>query-source</code> currently applies only to UDP queries; TCP queries always use a wildcard IP address and a random unprivileged port.</p>
<pre>recursion ( yes   no );</pre>	<p>If <code>yes</code> (default), the server attempts to do all the work required to answer a query that has requested recursion. Turning this off results in the server responding to the client with referrals.</p> <p>To prevent the server's cache from growing, use <code>recursion no</code> in combination with <code>fetch-glue-no</code>.</p>
<pre>rrset-order {order_spec; [order_spec; ... ] }</pre>	<p>See the <i>Resource Record Sorting</i> section.</p>
<pre>sortlist {address_match_list; };</pre>	<p>See the <i>Resource Record Sorting</i> section.</p>
<pre>topology {address_match_list; };</pre>	<p>Not implemented for BIND 9</p>
<pre>transfer-format <i>number</i></pre>	<p>The server supports two zone transfer methods. <code>one-answer</code> uses one DNS message per resource record transferred. <code>many-answers</code> packs as many resource records as possible into a message. <code>many-answers</code> is more efficient, but is only known to be understood by BIND 8.1 and higher and patched versions of BIND 4.9.5. The default is <code>one-answer</code>. <code>transfer-format</code> may be overridden on a per-server basis by using the <code>server</code> statement.</p>

<pre>transfer-source ip_addr;</pre> <p>see also <i>transfer-source-v6</i></p>	<p>Determines which local address will be bound to the TCP connection used to fetch all zones transferred inbound by the server.</p> <p>If not set, the value defaults to a system controlled value, usually the address of the interface “closest to” the remote end. This address must appear in the remote end’s <code>allow-transfer</code> option for the zone being transferred, if one is specified.</p> <p>This statement sets the <code>transfer-source</code> for all zones, but can be overridden on a per-zone basis by including a <code>transfer-source</code> statement within the zone block in the configuration file.</p>
<pre>transfers-in number</pre>	<p>The maximum number of inbound zone transfers that can be running concurrently. The default value is 10. Increasing <code>transfers-in</code> may speed up the convergence of slave zones, but it also may increase the load on the local system.</p>
<pre>transfers-per-ns number</pre>	<p>The maximum number of inbound zone transfers (<code>named-xfer</code> processes) that can be concurrently transferred from a given remote nameserver. The default value is 2. Increasing <code>transfers-per-ns</code> may speed up the convergence of slave zones, but it also may increase the load on the remote nameserver. <code>transfers-per-ns</code> may be overridden on a per-server basis by using the <code>transfers</code> phrase of the <code>server</code> statement.</p>
<pre>version version_string;</pre>	<p>Specifies the version number the server should report via the <code>ndc</code> command or via a query of name <code>version.bind</code> in class <code>chaos</code>. The default is the real version number of the server.</p>

# Address\_match\_list

The following can be address match lists:

- An IP address (in dotted-decimal notation)
- Another address match list
- An IP prefix (in /- notation)
- An address match list defined with the `acl` statement
- A key ID, as defined by the `key` statement

The following Access Control Lists (ACLs) are predefined and are not case-sensitive:

- any
- none
- localhost
- localnets

Place the `!` character in front of elements you want to negate.

Remember that address match lists follow the standard `named.conf` syntax and require a semi-colon (`;`) after each element. For example:

```
allow-update { !192.168.0.1; 192.168.0.0/16; };
```

When an IP address or prefix is compared to an address match list, the list is examined and the first match (regardless of its negated state) is used. The interpretation of a match depends on the conditions defined in the following table.

When a list is being used...	A non-negated match...	A negated match...
as an access control list	allows access.	denies access.
with the <code>listen-on</code> option	causes the DNS server to listen on matching interfaces.	causes the DNS server to NOT listen on matching interfaces.
with the <code>topology</code> clause	returns a distance based on its position on the list; the closer the match to the start of the list, the shorter the distance between it and the server.	is assigned the maximum distance from the server.



		<b>Note:</b> If there is no match, the address gets a distance that is further than any non-negated list element, and closer than any negated element.
--	--	--

Since the address match list uses a first-match algorithm, care must be taken when using negation. In general, if an element is a subset of another element, the subset should be present in the list before the broader element.

For example, `10.0.0/24; !10.0.0.1` will never negate to the `10.0.0.1` address because a `10.0.0.1` address will match with the `10.0.0/24` element and not traverse any farther. So the `10.0.0.1` address will be accepted in the match list.

Using `!10.0.0.1; 10.0.0/24` will elicit the desired effect. The `10.0.0.1` will be matched against the first, negated, element. All other `10.0.0.*` addresses will pass by the `10.0.0.1` element and be matched against the `10.0.0/24` subnet element.

## Logging

The `logging` section configures a wide variety of logging options for the nameserver. Its channel phrase associates output methods, format options and severity levels with a name that can be used with the category phrase to select how various classes of messages are logged. The basic `logging` syntax is as follows:

```
logging {
    channel channel_name {
        file name [versions number] [size bytes];
        | syslog daemon;
        | null;
        severity type;
        print-category yes_or_no;
        print-severity yes_or_no;
        print-time yes_or_no;
    };
    category category_name {
        channel_name; [ channel_name; ... ]
    }
}
```

```
};
};
```

Only one logging section is used to define as many channels and categories as you want. If there are multiple logging sections in a configuration, the first one defined determines the logging, and warnings are issued for the others. If there is no logging section, the default logging configuration will be:

```
logging {
  category default { default_syslog; default_debug; };
  category panic { default_syslog; default_stderr; };
  category packet { default_debug; };
  category eventlib { default_debug; };
};
```

The following is an example:

```
logging {
  channel xfers {
    file "MULTINET:XFERS.LOG";
    severity info;
    print-severity yes;
    print-time yes;
  };
  category xfer-in {
    xfers;
  };
};
```

This table describes the logging options.

Options	Description																								
channel	Specifies where the logging data goes: to syslog (OPCOM), to a file, to stderr (SYS\$ERROR), or to null (discarded).																								
category	<p>Specifies what data is logged. You can send a category to one channel or to many channels. These are the valid categories:</p> <table data-bbox="435 1564 1404 1766"> <tbody> <tr> <td>cname</td> <td>lame-servers</td> <td>packet</td> <td>security</td> </tr> <tr> <td>config</td> <td>load</td> <td>panic</td> <td>statistics</td> </tr> <tr> <td>db</td> <td>maintenance</td> <td>parser</td> <td>update</td> </tr> <tr> <td>default</td> <td>ncache</td> <td>queries</td> <td>xfer-in</td> </tr> <tr> <td>eventlib</td> <td>notify</td> <td>response-checks</td> <td>xfer-out</td> </tr> <tr> <td>insist</td> <td>os</td> <td></td> <td></td> </tr> </tbody> </table>	cname	lame-servers	packet	security	config	load	panic	statistics	db	maintenance	parser	update	default	ncache	queries	xfer-in	eventlib	notify	response-checks	xfer-out	insist	os		
cname	lame-servers	packet	security																						
config	load	panic	statistics																						
db	maintenance	parser	update																						
default	ncache	queries	xfer-in																						
eventlib	notify	response-checks	xfer-out																						
insist	os																								

<code>file</code>	Specifies the path name of the file you want the message to go to, and if you want to have multiple versions of the file, and if you want to limit the size of the file.
<code>syslog daemon</code>	Specifies that the message goes to syslog (OPCOM) instead of to a file.
<code>severity</code>	Specifies the severity level for this channel. The severity choices are: <code>critical</code> , <code>error</code> , <code>warning</code> , <code>notice</code> , <code>info</code> , <code>debug [level]</code> , and <code>dynamic</code> .
<code>print-category</code> <code>print-severity</code> <code>print-time</code>	Specifies whether to print the category, severity level, and time stamp of the messages. The default is NO for each item.

## Resource Record Sorting

When returning multiple resource records (RRs), the nameserver returns them by default in round robin order, that is, after each request the first RR is placed at the end of the list. You can specify in the `NAMED.CONF` file that the nameserver should sort the RRs based on the client's address using the `sortlist` option, or you can use a default other than round-robin using the `rrset-order` option.

The `sortlist` option sorts the RRs based upon the `address_match_list`. Each top level statement in the address match list must be an address match list with one or two elements. The first element of each address match list is checked against the client's address until a match is found. When a match is found:

<b>If the top level statement contains...</b>	<b>Then...</b>
only one element	that network is moved to the front of the list.
two elements	the second element is treated like the address match list in the <code>topology</code> option (see <i>NAMED.CONF Options</i> ).

Use the following `sortlist` statement to have the nameserver behave like the BIND 4.9.x nameserver.

Responses to...	
queries from the local host	prefers any of the directly connected networks
queries from any other hosts on a directly connected network	prefers addresses on that same network
other queries	are not sorted

```
sortlist {
  { localhost; localnets; };
  { localnets; };
};
```

The `rrset-order` option permits configuration of the ordering for the records in a multiple record response. An `order_spec` is defined as follows:

```
[ class class-name ] [ type type-name ] [ name "fqdn" ] order ordering;
```

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 some random order
cyclic	records are returned in round-robin order (default)

The following example specifies that only A, NS, and MX records are round-robin. This provides the same behavior as the MultiNet 4.2 nameserver.

```
rrset-order {
  class IN type A name "*" order cyclic;
  class IN type NS name "*" order cyclic;
  class IN type MX name "*" order cyclic;
  order fixed;
};
```

# Incremental Zone Transfer

BIND 9.7.2-p3 contains an implementation of incremental zone transfer (IXFR) - it is on by default. If you need to turn it off for a particular slave server, use the `provide-ixfr` `server` substatement, which defaults to `yes`:

```
server ip_addr {
    provide-ixfr ( yes | no );
};
```

# DNS Zone Information Files

The DNS server configuration file contains references to DNS zone information files, that contain control information and a list of resource records for objects in the zone. The file format of a zone information file is:

```
$INCLUDE filename [domain]
$ORIGIN domain
$TTL default-ttl
$GENERATE range lhs type rhs
domain [ttl] [class] record-type resource_data
```

If the `domain` is specified as `"."`, the domain is the `ROOT` domain. If the `domain` is specified as an `@` sign, the domain is the current origin. Anything else is taken as a standard domain name, that if terminated by a dot (`.`) is used verbatim; otherwise, the current origin is appended to the specified domain name.

The optional `domain` field in a `$INCLUDE` line is used to define an origin for the data in an included file. It is equivalent to placing a `$ORIGIN` statement before the first line of the included file. The field is optional. Neither the optional `domain` field in the `$INCLUDE` line nor `$ORIGIN` statements in the included file modify the current origin for this file.

The `$TTL` statement sets the default time-to-live for records that do not have explicit `ttl` fields. If the zone file does not have a `$TTL` statement, the DNS server prints a warning on your computer screen and uses the minimum value from the `SOA` record.

The optional `ttl` field is an integer value to specify in the time-to-live field in the following ways:

Each of these is equivalent to one week.

- 604800

- 1w
- 7d
- 168h
- 10080m
- or any combination

For example: `sigma 2h46m40s IN A 192.1.1.97`

Loads the TTL as: `t1 = 10000` (2 hours 46 mins 40 secs)

The default is to use the value specified in the `$TTL` directive or SOA resource record for the zone. The optional class field is the object address type; currently only one type (IN, for objects connected to the Internet) is supported.

The `record-type` field is also known as "resource record types." The below table shows the most commonly used DNS resource record types and their uses (the data expected in the `resource_data` field is shown in brackets [ ]).

#### DNS Resource Record Types:

Record Type	Use
A	A host address [IP-address]
CNAME	The canonical name for an alias [domain-name]
HINFO	Host information [CPU-type OS-type]
KEY	The public key associated with a domain name [flags protocol algorithm key]
MB	A mailbox domain name [domain-name]
MG	A mail group member [domain-name]
MINFO	Mailbox or mail list information [request-domain error-domain] <sup>b</sup>
MR	A mail rename domain name [domain-name]
MX	A mail exchanger [preference domain-name]

NS	An authoritative name server [domain-name]
NULL	A null resource record [none]
NXT	Used for secure negative responses. Tells a querier which record is lexicographically next in the zone [next-domain-name type-bitmap]
PTR	A domain name pointer [domain-name]
SIG	A security signature for an RRset [type algorithm labels ttl expiration inception tag name signature]
SOA	The start of a zone of authority [domain of originating host, domain address of maintainer, a serial number and the following parameters in seconds: refresh, retry, expire, and minimum time-to-live] (see RFC-1035)
SRV	Specifies the location of services [priority weight port target]
TXT	Arbitrary text [text-string]
WKS	A well-known service description [address protocol service-list]

Resource records are usually single-line entries, but SOA records may be continued across lines by surrounding the resource record statements with open and close parentheses. Comments begin with semicolons and continue to the end of the line. Each zone information file should begin with an SOA resource record for the zone. An example SOA resource record follows.

```
@      IN      SOA   VMSITE.EXAMPLE.COM.      system.EMAIL.EXAMPLE.COM. (
                                2000022101      ; serial number as yyyymmddnn
                                7200             ; refresh every 2 hours
                                7200             ; retry every 2 hours
                                12096000        ; expire in twenty weeks
                                86400 )         ; minimum time-to-live
```

The SOA resource record lists a serial number that DNS administrators should increase each time they modify the master file. Secondary servers check the serial number at intervals specified by the refresh time. If the serial number has increased since the last zone transfer, the secondary name server requests a new zone transfer and then loads the new zone data.

In the preceding example, VMSITE is a host in the EXAMPLE.COM domain. This should be the primary DNS server for this zone.

"system.EMAIL.EXAMPLE.COM" is the email address of the DNS zone administrator on the EMAIL host in the EXAMPLE.COM domain.

**Note:** Although the DNS zone administrator's email address is specified without an at sign (@), the effective email address requires changing the first period to @. For example, the email address of the DNS zone administrator in the preceding example is system@EMAIL.EXAMPLE.COM.

If you configure your name server as a secondary name server, it contacts the primary name server for a new zone transfer after the refresh interval expires. If the server does not receive a response after the "retry" interval, it tries repeatedly to contact the primary name server until it succeeds. If the secondary server fails to contact the primary name server before the expire interval elapses, it discards all data from the zone.

The minimum value is the time-to-live value used by records in the file with no explicit value if there is no \$TTL statement. It is used also as the time-to-live value for negative caching.

Set the expire time to a value long enough to accommodate the retry and refresh intervals. If the refresh interval exceeds the expiration time, the data on your secondary server will expire before new data can be loaded. The following example shows a zone information file for the zone EXAMPLE.COM.

```
;
;   Authoritative data for EXAMPLE.COM
;
$TTL 86400
@   IN   SOA  VMSITE.EXAMPLE.COM. system.EMAIL.EXAMPLE.COM. (
                                2000022107 ; serial number as yyyyymmddss, where
                                ; ss is the zone change sequence
count
                                7200      ; refresh every 2 hours
                                7200      ; retry every 2 hours
                                12096000 ; expire in twenty weeks
                                86400 )  ; minimum time-to-live
                                IN       NS   VMSITE.EXAMPLE.COM.
                                IN       NS   SPACELY.EXAMPLE.COM.
;
;   Information about the host EXAMPLE.COM
;
EXAMPLE.COM.  IN       MX     0          EXAMPLE.COM.
              IN       A      192.0.0.1
```



```

                IN      HINFO   VAXSTATION-9440 VMS
;
;   The loopback address and host
;
LOCALHOST.     IN      A       192.0.0.1
                IN      HINFO   LOOPBACK-HOST LOOPBACK
;
;   The remaining hosts
;
AARDVARK       IN      A       192.0.0.2
                IN      MX      0         AARDVARK
                IN      HINFO   VAXSTATION-3200 VMS
John           IN      A       192.0.0.3
                IN      MX      0         John.EXAMPLE.COM.
                IN      HINFO   VAXSTATION-3100 VMS

```

The SOA resource record indicates the start of authority for the zone EXAMPLE.COM. The NS resource records indicate which DNS name servers are authoritative for the zone. The remainder of the file lists each host and information about it. VMSITE is a system in the EXAMPLE.COM domain. The system.EMAIL.EXAMPLE.COM is the email address on the EMAIL host in the EXAMPLE.COM domain.

An example of a zone information file for the zone 0.128.IN-ADDR.ARPA follows. This file contains the information needed to map IP addresses in the network 128.0.0.0 into host names. This file contains an example of the PTR record type.

```

;
;   Authoritative data for 0.128.IN-ADDR.ARPA
;
$TTL 86400
@      IN      SOA   VMSITE.EXAMPLE.COM.   system.EMAIL.EXAMPLE.COM. (
                                2000022101 ; serial number as yyymmddnn
                                7200        ; refresh every 2 hours
                                7200        ; retry every 2 hours
                                12096000   ; expire in twenty weeks
                                86400 )    ; minimum time-to-live
                IN      NS      VMSITE.EXAMPLE.COM.
                IN      NS      SPACELY.EXAMPLE.COM.
;
; Network database
;
1.0    IN      PTR    EXAMPLE.COM.
2.0    IN      PTR    AARDVARK.EXAMPLE.COM.
3.0    IN      PTR    JOHN.EXAMPLE.COM.

```

# Reloading the Name Server

To reload the running server:

```
$ MULTINET NETCONTROL DOMAIN RELOAD
```

To restart the name server:

```
$ MULTINET NETCONTROL DOMAIN RESTART
```

**Note:** The master server must be restarted before the nameserver restarts to load in any `net-config` or `server-config` changes `@multinet:start_server`.

# Controlling the MultiNet DNS Server

You can use the MultiNet NETCONTROL server to request that the DNS server perform specific actions. The name to specify to NETCONTROL for the DNS server is DOMAINNAME.

Refer to the NETCONTROL DOMAINNAME table of commands in the *MultiNet Administrator's Reference*.

When the server is busy, NETCONTROL sends a message stating that your request has been queued, and it will be acted upon when it is the next one in the queue to be serviced. When the server is not busy, it performs your request while NETCONTROL waits (except for the case of RELOAD). For example,

```
DOMAINNAME>stat
<Dumping Nameserver Statistics
<Domain-Name-Server Busy, Request Queued
```

# Using NSLOOKUP and DIG to Debug DNS

The MultiNet NSLOOKUP and DIG utilities send test queries to a DNS Name Server to test the configuration. NSLOOKUP enters interactive mode when invoked without arguments. The table below describes the valid interactive commands. For information about DIG and more information about NSLOOKUP, see Chapter 1 of the *MultiNet Administrator's Reference*.

Command	Description
---------	-------------

name	Prints information about name using the default server.
name server	Prints information about name using server.
exit	Exits NSLOOKUP.
finger [user]	Finger the optional user at the current default host.
help or ?	Prints help information.
set all	Prints the current status of all options.
set class=class	Sets the query class to one of these: IN, CHAOS, HESIOD, or ANY.
set [no]debug	Prints debugging information.
set [no]d2	Prints exhaustive debugging information.
set [no]defname	Appends the domain name to each query.
set [no]recurse	Asks for a recursive answer to query.
set [no]vc	Always uses a virtual circuit (TCP instead of UDP).
set domain=name	Sets the default domain name to name.
set port=port	Sets the port number on which to send a query.
set root=name	Sets the root name server to name.
set retry=n	Sets the number of retries to n.
set srchlist=name1 [/name2/.../name6]	Sets the domain to name1 and the search list to name1 through name6.
set timeout=n	Sets the timeout interval to n.

<pre>set querytype=type or set type=type</pre>	Sets the resource record (RR) type to query for.
<pre>server name</pre>	Sets the default server to name, using the current default server.
<pre>lserver name</pre>	Sets the default server to name, using the original default server.
<pre>root</pre>	Sets the current default server to the root.
<pre>ls [option] name [&gt; file]</pre>	<p>Lists the domain name, with output optionally going to file. option is one of the following:</p> <ul style="list-style-type: none"> <li>-a List canonical names and aliases</li> <li>-h List HINFO (CPU type and operating system)</li> <li>-s List well-known services</li> <li>-d List all records</li> <li>-t type List records of the given type (such as A, CNAME, and MX)</li> </ul>

## DNS Load Balancing

The MultiNet domain name server provides a feature called DNS load balancing, which is modeled after the service names available with HP's LAT terminal server support for VMScluster systems. The domain name server maintains a load rating for each node offering a particular service name and, when queried for the addresses records for that name, it orders them based on the load rating. This allows TCP-based services such as TELNET and FTP to be offered cluster-wide in a load-balanced fashion. To configure DNS load balancing on each cluster node that will offer the service:

1. Set up cluster services offered by this node with NET-CONFIG (see the *Setting Up a Cluster Service* section).
2. If your host is multi-homed (one that has more than one IP network interface), specify the IP address associated with the cluster services with NET-CONFIG (see the *Advertised Cluster Service Addresses on Multi-Homed Hosts* section).
3. Configure service ratings for advertised services (see the *Setting Service Ratings* section).

4. Add the new service names to your domain's DNS zone file (see the *Adding Cluster Services to Your Domain's DNS Zone File* section).
5. Monitor and test the status of your cluster service names with the `MULTINET NETCONTROL DOMAINNAME SHOW` command (see the *Monitoring Cluster Service Names* section).

## Setting Up a Cluster Service

To configure a cluster service name with `NET-CONFIG`:

```
$ MULTINET CONFIGURE  
NET-CONFIG>SET CLUSTER-SERVICE-NAMES name[,...]
```

**Note:** Don't confuse the DNS load balancing feature and the `CLUSTER-SERVICE-NAMES` parameter with the cluster alias feature and `IP-CLUSTER-ALIAS` parameter described in Chapter 11.

Each cluster service name must be a name *not already in use* on your network. Specify each name in its fully qualified form (for example, `CLUSTER.EXAMPLE.COM`). If you configure more than one name, separate the names with commas when you specify them. To activate your cluster service names on the running system, use the command:

```
$ DEFINE/SYSTEM/EXEC MULTINET CLUSTER SERVICE NAMES "name"
```

If you are setting up multiple service names, enclose each name in quotation marks and separate the quoted names with commas. Once the logical name has been defined, restart the nameserver:

```
$ MULTINET NETCONTROL DOMAINNAME RESTART
```

## Advertised Cluster Service Addresses on Multi-Homed Hosts

For a multi-homed host, you can control the address advertised for a cluster service with the `NET-CONFIG SET CLUSTER-SERVICE-ADDRESS` command:

```
NET-CONFIG>SET CLUSTER-SERVICE-ADDRESS address
```

If you do not configure a cluster service address for a host with multiple interfaces, the address advertised for the cluster service will be selected at random.

## Setting Service Ratings

The load-rating algorithm used by the DNS server for cluster service names is similar to the algorithm used by LAT. It is based on the system load, the number of interactive processes on the system, and the amount of free physical memory on the system. If your VMSccluster system contains CPUs of vastly different speeds and/or memory configurations, you may find that the algorithm always favors a faster or larger-memory system over a slower or smaller-memory system.

In an unbalanced cluster configuration, you may need to set either a *static load rating* or weight the rating computation by setting a *CPU rating* on each node:

- Static Ratings - A static rating is set by defining a logical name:

```
$ DEFINE/SYSTEM/EXEC MULTINET CLUSTER SERVICE STATIC_RATING n
```

*n* is a decimal number ranging from 1 to 255.

When defined, this rating will be used in all cluster service advertisements, bypassing the dynamic load rating computations. The higher the number you set, the more "available" (or less loaded) the system will be.

- CPU Ratings - A CPU rating is also set by defining a logical name:

```
$ DEFINE/SYSTEM/EXEC MULTINET CLUSTER SERVICE CPU_RATING n
```

*n* is a decimal number ranging from 1 to 100.

This is the weight value factored into the load rating calculation. To bias load ratings so that faster CPUs service more users, set a lower CPU rating value on your slower CPUs and a higher CPU rating on your faster CPUs.

After setting ratings, restart the nameserver:

```
$ MULTINET NETCONTROL DOMAINNAME RESTART
```

## Adding Cluster Services to Your Domain's DNS Zone File

Once you have configured a cluster service name on your cluster, update your domain's primary name server DNS zone file to delegate authority over the cluster service name to the participating cluster nodes. To do this, add NS records that map each cluster service name to the participating nodes. For

example, to add the cluster service name CLUSTER.EXAMPLE.COM, add the following lines to the configuration files on EXAMPLE.COM's primary name server:

```
CLUSTER.EXAMPLE.COM.  IN  NS  NODE1.EXAMPLE.COM.  
CLUSTER.EXAMPLE.COM.  IN  NS  NODE2.EXAMPLE.COM.
```

The name on the left side is the cluster service name; the name on the right side is the domain name of a node offering the cluster service.

## Monitoring Cluster Service Names

To check the status of your cluster service names, use the following NETCONTROL command:

```
$ MULTINET NETCONTROL DOMAINNAME SHOW
```

For each cluster service name, a listing of the nodes offering the service and their load ratings is displayed. For example:

```
$ MULTINET NETCONTROL DOMAINNAME SHOW
```

```
Connected to NETCONTROL server on "NODE1"  
< Node1.Example.COM Network Control V5.6(10) at Mon 13-Mar-2020 4:35PM-EST  
< Service CLUSTER.EXAMPLE.COM:  
<  Nodename          Address          Rating  
<  -----          -  
<  NODE1             10.41.228.101   75  
<  NODE2             10.41.228.102   83  
< End of line
```

You should also test a cluster service name using NSLOOKUP:

```
$ MULTINET NSLOOKUP CLUSTER.EXAMPLE.COM
```

```
Server:    LOCALHOST  
Address:   10.0.0.1  
Name:     CLUSTER.EXAMPLE.COM  
Addresses:10.44.128.102, 10.44.128.101
```

NSLOOKUP should return the addresses from highest to lowest rating, although DNS caching can cause address ordering to lag behind rating changes for short periods of time.

# DNS Security

BIND 9 includes an implementation of DNS Security (DNSSEC). A complete description of DNSSEC and its use is beyond the scope of this chapter. DNSSEC is described in RFC 2065 "Domain Name System Security Extensions" and various internet drafts.

MultiNet includes the following BIND tools related to DNS Security:

- DNSKEYGEN – Used for generating and maintaining keys.
- DNSSIGNER - Used for signing zones.

For more information on these tools, see the *MultiNet Administrator's Reference*.

This section describes a simple scenario in which DNSKEYGEN and DNSSIGNER are used.

The simplest "normal" case is a zone which has no delegations, and is to be signed with a single zone key. Assume that the parent zone is secured and is able to sign the public zone key.

The first step in signing a zone is to generate a private-public key pair. This is done using DNSKEYGEN. This will generate a DNS zone master file version of the public key in a file with the suffix "key".

## Example of key generation:

```
$ MULTINET DNSKEYGEN/DSA=768/ZONE/NOENCRYPT ZZ.TEST.
```

This results in the generation of two files, the names of which reflect the key owner, algorithm, and footprint. The names end in "key" and "private". The "key" file contains the DNS RR holding the public key, the "private" file has the data defining the private key. The latter file is set to be read/write only by the file's owner.

## Example key file (key represented in base64 characters):

```
zz.test. IN KEY 16641 3 3 AQPIc...
```

## Example private file:

```
Private-key-format: v1.1  
Algorithm: 3 (DSA)  
Prime(p): base 64 characters  
Subprime(q): base 64 characters  
Base(g): base 64 characters  
Private_value(x): base 64 characters  
Public_value(y): base 64 characters
```



**Note:** The two numeric fields in the key filenames will be different for each time `dnskeygen` is run. Also note that the "private" key's format will depend on the algorithm used to derive the key.

The next step is to run `DNSSIGNER` over the data. To make things simple, all files involved will be considered to be in the current default directory unless otherwise stated. In the directory where the file `zone.1` resides there should be a "private" file for the key used for signing.

### Example zone file (zone.1):

```
$ORIGIN zz.test.
@      IN   SOA  a.test. a.a.test. 1 360 36 60480 12
        NS  a.test.
        NS  b.test.
one    A   10.10.10.10
two    A   10.10.10.100
        MX  10 one.zz.test.
a.test. A   10.11.12.13
b.test. A   10.13.12.11
```

The public key (from the key file) is sent two different ways. One copy of the public key is sent to the parent zone for signing with the parent's zone key. The public key is also copied (or even `$INCLUDE'd`) into the `zone.1` file. Signing may begin prior to receiving a response from the parent zone (which contains, among other things, the signed public key).

Although the public key is going to arrive from the parent at some time packaged with the signature, the unsigned key must be placed into the unsigned zone master file. The presence of the public key record alerts `DNSSIGNER` to perform certain functions, such as generating `NXT` records and generating parent files for its delegation points.

**Caution!** Although `DNSSIGNER` is flexible enough to withstand missing private keys, and late arriving parent files, it cannot be expected to behave correctly if the data used to derive the zone master file changes during the execution of the signing process. In accordance with this, the public key should be added to the zone even though the key will also arrive from the parent later. `DNSSIGNER` will remove duplicate records.

### Example of signing a zone:

```
$ MULTINET DNSSIGNER/ZONE=(INP=ZONE.1, out=ZONE.2) -  
/SIG=KEY=(DOM=ZZ.TEST, ALG=3, KEY_ID=6750)
```

The result of the run will be a new zone file. The file `zone.2` will appear something like the following:

### Example output of DNSSIGNER (zone.2):

```
$ORIGIN zz.test.  
zz.test. 12      IN      SOA      a.test. a.a.test. 1 6M 36S 16h48m 12S  
          SIG      SOA 1 12 19980223163147 19980123163147 6750 zz.test. (...)  
zz.test. KEY      0x4101 3 1 (...)  
zz.test. NS       a.test.  
          NS       b.test.  
          SIG      NS 1 12 19980223163147 19980123163147 6750 zz.test. (...)  
zz.test. NXT      one.zz.test. NS SOA SIG KEY NXT  
          SIG      NXT 1 12 19980223163147 19980123163147 6750 zz.test. (...)  
one       A       10.10.10.10  
          SIG      A 1 12 19980223163147 19980123163147 6750 zz.test. (...)  
one       NXT      two.zz.test. A SIG NXT  
          SIG      NXT 1 12 19980223163147 19980123163147 6750 zz.test. (...)  
two       A       10.10.10.100  
          SIG      A 1 12 19980223163147 19980123163147 6750 zz.test. (...)  
two       MX       10 one.zz.test.  
          SIG      MX 1 12 19980223163147 19980123163147 6750 zz.test. (...)  
two       NXT      zz.test. A MX SIG NXT  
          SIG      NXT 1 12 19980223163147 19980123163147 6750 zz.test. (...)  
a.test.  A       10.11.12.13  
b.test.  A       10.13.12.11
```

All of the "(...)" fields are base64 encoded values. This file is complete except for the missing signature by `test.` over the `zz.test. KEY` record. If this file is sent to a secured name server, the zone data will be rejected unless the zone key happens to have been configured.

**Note:** It is wise not to configure the zone key for a zone unless the parent will not be signing the zone key.

Eventually, the parent file will arrive. After obtaining the file, the DNSSIGNER needs to be run again to include the new data.

### Example parent file (parent.1):

```
zz.test. NXT      zzz.test. NS SIG KEY NXT  
zz.test. SIG      NXT 1 12 19980229163147 19980129163147 12345 test. (...)  
zz.test. KEY      0x401 3 1 (...)  
zz.test. SIG      KEY 1 12 19980229163147 19980129163147 12345 test. (...)
```

The final run of DNSSIGNER is:

```
$ MULTINET DNSSIGNER/ZONE=(IN=ZONE.2,OUT=ZONE.3)/PARENT=IN=PARENT.1
```

**Note:** The specification of the key is no longer needed. However, now that the records are signed, DNSSIGNER will verify all the existing signatures.

In the case that a signature fails during these checks, the action taken by DNSSIGNER depends on whether the key of the signature is specified on the DNSSIGNER command line during the run. In the example, failing signatures are just dropped. If the run command included

```
/SIG=KEY=(DOM=ZZ.TEST.,ALG=DSA,KEY_ID=6750)
```

then failing signatures would be replaced.

The result of the second run of DNSSIGNER is `zone.3`, which is the final zone file and would be used by the nameserver. `zone.3` is a merger of `zone.2` and `parent.1`, minus the records which appear in both files; that is, duplicates are removed.

## Multicast Name Resolution

Multicast name resolution aims to eliminate the need to maintain `HOSTS` files or configure a name server on networks that are contained within a single logical LAN. Systems participate by sending out a multicast request to resolve a name and any system that recognize the name responds to the request. Systems that participate in multicast name resolution use one of two protocols: LLMNR (RFC 4795), or mDNS. Both protocols use packets that are very similar to standard DNS packets; they operate in different multicast groups and use different port numbers.

MultiNet offers a responder that participates in both protocols and the ability to configure the resolver to use one of the two protocols. Using a multicast group disables one of the resolver's checks for authenticity of the answers that it receives. The multicast name responder works for both IPv4 and IPv6 addresses. Zero configuration of systems is one of the goals for IPv6 on small networks, and multicast name resolution helps in meeting this goal. Configuring MultiNet to use multicast name resolution involves enabling the server (LLMNR) and setting the name server address and port with `MULTINET CONFIGURE/NETWORK`.

### Configuring Multicast Name Resolution

```
$ multinet configure/server
```

```
SERVER-CONFIG>ENABLE LLNMR  
SERVER-CONFIG>WRITE  
SERVER-CONFIG>EXIT
```

```
$ multinet configure/network  
NET-CONFIG>set multicast-name-resolution {LLNMR | mDNS}  
NET-CONFIG>WRITE  
NET-CONFIG>EXIT
```

LLNMR (port 5355 on 224.0.0.252 and FF02::1:3)  
mDNS (port 5353 on 224.0.0.251 and FF02::FB)

```
$ define/system/exec/nolog multinet_nameservers 224.0.0.252  
$ define/system/exec/nolog multinet_dns_port 5355
```

# 8. Establishing IP Connectivity

This chapter explains how to establish IP connectivity between your computer and other computers on your network. Connectivity depends upon the network interfaces in your system.

## About IP Connectivity

Establishing IP connectivity ensures that users can perform the tasks described in the *MultiNet User's Guide*, including:

- Obtaining information about remote systems and users with FINGER and WHOIS
- Accessing files on other computers with the FTP, RCP, TFTP, SCP, and SFTP commands
- Logging into other computers with the RLOGIN, TELNET, and SSH commands
- Using remote printers

**Note:** Establishing IP connectivity only ensures you can reach other systems if you know their IP addresses. It does not ensure you can reach other systems by name. For example, there is no guarantee you can send mail to users on remote systems without first configuring host tables or the Domain Name System (DNS) (see Chapter 10).

## Network Interface Configuration Overview

At startup, MultiNet obtains global configuration data (such as the default route) and device-specific network interface configuration data (such as the IP address) from the following files:

- The `MULTINET:NETWORK_DEVICES.CONFIGURATION` network database file describes the current network configuration, including a list of the device interfaces you have specified. This file is used to determine which devices are present when the network is started.
- The `MULTINET:START_MULTINET.COM` network initialization command procedure starts and configures the network, and initializes individual device interfaces and global parameters. This file is overwritten each time you use `NET-CONFIG` to update and save the configuration.

This chapter describes how to modify the startup command procedure and configuration file with `NET-CONFIG` and related steps required to ensure successful configuration. For details on using these configuration utilities, see Chapter 9.

**Note:** Network interface configuration changes take effect the next time your system reboots. In contrast, most global parameter changes do not require rebooting.

Some network interfaces require configuration data that is not accessible through either `NET-CONFIG`. When configuring such interfaces, additional configuration files are required. For details, see the “Creating a Custom Interface Initialization Procedure” section.

## Supported Network Interface Devices

MultiNet allows you to configure multiple interface devices on your network. Each interface is named according to its type (for example, shared Ethernet interfaces are of the type "se").

In general, MultiNet accommodates a maximum of ten devices of each type. This table lists the devices MultiNet supports and the interface names to use when specifying devices to be added or modified.

Device	Interface Name
Raw packet (dbridge)	<code>rp[0...49]</code>
SLIP (Serial line IP) using any VMS-supported terminal multiplexer	<code>sl[0...49]</code>

PPP (Point-to-Point Protocol)	ppp[0...49]
HP VMS Ethernet, FDDI, ATM, or Alpha Token-Ring controller	se[0...19]
STF (IPv6 encapsulated in IPv4)	stf0

MultiNet contains a driver for each of these device types. The driver either accesses the device directly or is an interface to an appropriate OpenVMS device driver.

## Viewing Interface Configuration

You can use the `NET-CONFIG SHOW` command to display the maximum configuration or the current configuration.

### Viewing the Maximum Configuration

Use the following command to display all interface types supported by MultiNet are displayed, including the default settings for the Adapter, CSR, Flags, and Vector parameters:

```
$ MULTINET CONFIGURE /INTERFACE
NET-CONFIG>SHOW MAXIMUM
```

For example:

```
NET-CONFIG>SHOW MAXIMUM
Devices                               Adapter  CSR  Address  FlagsVector
-----                               -
rp[0-9]   (Raw Packet)                -NONE-  -NONE-  -NONE-
ppp[0-49] (Point-to-Point Protocol)  -NONE-  -NONE-  -NONE-
se[0-9]   (Shared VMS Ethernet/FDDI) -NONE-  -NONE-  -NONE-
sl[0-49]  (Serial Line IP)                  -NONE-  -NONE-  -NONE-
```

### Viewing the Current Configuration

Use the following command to display global parameter settings and device interfaces currently in your network configuration (including the actual settings for Adapter, CSR, Flags, and Vector):

```
NET-CONFIG>SHOW [CURRENT]
```

For example:

```
NET-CONFIG>SHOW
Interface                               Adapter  CSR Address  Flags/Vector
-----                               -
se0  (Shared VMS Ethernet/FDDI)         -NONE-    -NONE-      -NONE-
      [TCP/IP: 10.0.0.68, IP-SubNet: 255.255.255.0]
      [VMS Device: EZA0, Link Level: Ethernet]
ppp0 (Point-to-Point Protocol)         -NONE-    -NONE-      -NONE-
      [VMS Terminal: TTA0]
      [ACCM: 0x0, Authentication: None]
      [Protocol Compression: Off, Address and Control Field
      Compression: Off]
[Idle Timeout: 0, Configuration Timeout: 0]
[MRU: 0, ICMP Allowed: Yes]
[Configuration Retries: 0, Termination Retries: 0]
[TCP Header Compression: Disabled]
Official Host Name:                     SFO.EXAMPLE.COM
Default IP Route:                       10.0.0.129
Domain Nameserver:                     127.0.0.1
Timezone:                               EST
Default TFTP Directory:                 MULTINET_ROOT:[MULTINET.TFTP]
Anonymous FTP Directory:                ANONVILLE:[ANONYMOUS]
Load EXOS $QIO driver:                  TRUE
Load UCX $QIO driver:                   TRUE
Load PWIP (Pathworks) driver:          TRUE
```

The SHOW command with no parameters defaults to SHOW CURRENT.

## Adding Network Interfaces

To add a new network interface to your MultiNet configuration, you can use NET-CONFIG (see the *Adding Network Interfaces with NET-CONFIG* section).

For details about this utility, see Chapter 9.

The tables below list all of the network interface parameters used by MultiNet-supported interfaces, and the parameters required by each type of interface.

## Network Interface Parameters

The supported network devices can be classified into three categories that determine the parameters you enter when configuring the device:



- Hardware devices with which MultiNet communicates directly. For each of these devices, NET-CONFIG requires that you specify the CSR and the UNIBUS adapter into which the device is plugged. Most of the devices also require you to specify device-specific parameters.

Some of these devices have programmable interrupt vectors that you specify with NET-CONFIG; MultiNet programs these vectors during startup. Others have interrupt vectors that are determined by the hardware. For each of these devices, set the vector and the CSR on the hardware using the DIP switches or jumpers on the card as described in the device's manual. Each interrupt vector must be unique.

- Hardware devices through which MultiNet communicates using a VMS device driver. For these devices, NET-CONFIG requires that you identify the VMS device through which MultiNet is to communicate; for most of these devices, you must also specify device-specific parameters.
- Software, or pseudo, devices whose interfaces communicate with software and for which no hardware is directly associated. These interfaces are typically used to implement special-purpose transports and deliver packets to other software. For example, the IP-over-DECnet interface encapsulates IP packets in DECnet datagrams for transmission over a DECnet network. All parameters for these devices are device-specific.

This table lists the prompts that appear when you run NET-CONFIG. Make sure you respond to at least one network address prompt so the device can be started from the boot process. The following table lists the prompts displayed for each device type.

Parameter	Function
ACCM Mask	Asynchronous Control Character Map Mask. A 32-bit mask that indicates the set of ASCII control characters to be mapped into two-character sequences for transparent transmission over the line. Default is %x00000000.
Adapter	Identifies the UNIBUS to which the device is connected. The setting can be the name of a UNIBUS (UBA0, UBA1, UBA2, or UBA3), or ANY, which tells MultiNet to search each UNIBUS until it finds a device at the specified CSR.
Address and Control Field Compression (ACFC)	When ON, PPP eliminates the address and control fields when they are identical over a series of frames. Default is OFF.
Baud Rate	Indicates the transmission baud rate. Valid settings are 110, 300, 1200, 2400, 4800, 9600, 19200, and UNSPECIFIED.

BSD Trailer Encapsulation	For 10Mb/sec Ethernet controllers only. Can be used to enable Berkeley Trailer encapsulation of IP packets on those Ethernets. Two valid settings: <code>NEGOTIATED</code> or <code>DISABLED</code> (the default, which prevents the use of trailer encapsulation).
Communications Mode	For communications devices that share a dialup line with either a modem or a terminal. Use <code>DTE</code> (Data Transmit Enable, the default) to specify that the line can originate serial communications, or <code>DCE</code> (Data Carrier Enable) to specify the opposite.
CSR	Control Status Register. Identifies the device's octal bus address. The CSR is usually programmed by setting DIP switches or jumpers on the card as described in the device's documentation.
Flags	Some devices have a Flags prompt whose meaning is device-dependent.
Hardware Device	The name of the real Ethernet device; for example, <code>se0</code> .
Header Compression Mode	For SLIP devices, indicates whether to use Van Jacobson's header compression algorithm. The parameter has three valid settings:  <code>DISABLED</code> - Indicates that headers should never be compressed (default).  <code>ENABLED</code> - Indicates that headers should always be compressed.  <code>NEGOTIATED</code> - Indicates that headers should not be compressed until a compressed header is received from the other side.  At least one side of a link must be <code>ENABLED</code> for compression to be used; that is, both sides of a link cannot be set to <code>NEGOTIATED</code> .
ICMP	When <code>ENABLED</code> (the default), PPP allows ICMP packets over the PPP connection. Administrators may want to disable ICMP packets if they are concerned with "service attacks" from dial-up connections.
IP Address	Indicates the Internet address associated with the interface.

IP Address of Remote System	Indicates the Internet address of the system to which the interface will connect.
IP Broadcast Address	Used with devices that support broadcasts. Allows the setting of a non-standard IP broadcast address; defaults to an address with a host portion of all 1's.
IP Over DECnet Peer Host's DECnet Name	Used with IP-over-DECnet links to indicate the name of the DECnet node on the other end of the connection.
IP SubNet Mask	Allows setting a non-standard IP subnet mask.
IPv6 global address	Indicates the global unique address associated with this interface. The interface may also have a link-local address which will be automatically generated when the interface is started.
IPv6 mask length	The length of the mask for the IPv6 address.
Jumbo Frames	Used with Ethernet devices to indicate whether to use standard length Ethernet packets (1500 bytes) or larger (9000 bytes) Jumbo frames. Jumbo frames can provide a higher throughput rate because more data is processed on a single interrupt.
Link Level Encapsulation Mode	Used with Ethernet devices to indicate whether to use the standard Ethernet encapsulation of IP datagrams, or extended 802.2 encapsulation as specified in RFC1042. Valid settings are ETHERNET and 802.2.
Maximum Receive Unit (MRU) Size	Determines the maximum number of 8-bit bytes for the PPP Information field, including padding, but not including the Protocol field. Because opposite ends of a PPP connection may have different MRU values, PPP negotiates a suitable MRU for both systems. Default: 500.
Point-To-Point Device IP Destination Address	Used with point-to-point interfaces to indicate the IP address of the system on the other side of the line.
Protocol Compression	When ON, PPP negotiates with the peer to use one byte instead of two for the Protocol fields to improve transmission efficiency on low-speed lines. Default: OFF.

Retry Count	Determines the number of attempts PPP makes to configure a connection with "Configure-Request" packets. Default: 0.
Termination Retry Count	Determines the number of attempts PPP makes to terminate a connection with "Terminate-Request" packets. Default: 0.
Timeout	Determines the time (in seconds) between successive Configure-Request or Terminate-Request packets. Default: 0.
VMS Device	Used with devices that use a VMS device driver to interface to the hardware. Indicates the name of the VMS device that MultiNet is to use. This parameter is used with HP Ethernet interfaces and SLIP interfaces.
Vector	Used with programmable vector devices only. Identifies the interrupt vector that MultiNet assigns to the device during the boot process. Each interrupt vector (both fixed and programmable types) must be unique. Refer to the <i>Displaying Interrupt Vectors</i> section for an example of how to display the current system interrupt vectors.

**Note:** If your network requires a network interface to be initialized with parameters other than those listed in the above table, create a custom initialization command procedure as described in the *Creating a Custom Interface Initialization Procedure* section.

Type	Description
se	<p><b>Interface name:</b> se0, se1, se2, ... se19</p> <p><b>Device type:</b> Any HP VMS Ethernet, FDDI, ATM, or Alpha Token-Ring controller</p>

<b>Parameter Prompt</b>	<b>Example Value</b>
VMS Device:	XEA0
Link Level Encapsulation Mode:	ETHERNET
BSD Trailer Encapsulation:	DISABLED
IP Address:	10.0.0.70
IP SubNet Mask:	255.255.255.0
Non-Standard IP Broadcast Address:	10.0.0.71
DHCP CLIENT [DISABLED]:	DISABLED
Jumbo Frames [DISABLED]:	ENABLED
IPv6 on this interface [DISABLED]:	ENABLED
IPv6 global address [NONE]:	3FFE:1200:3006::C673:8EBE
IPv6 mask length:	48

The se interface uses any HP Ethernet controller to provide access to a 10/100/1000 Mb/s Ethernet network, any HP FDDI controller to provide access to a 100 Mb/s FDDI network, and any HP Alpha Token-Ring controller to provide access to 4 Mb/s or 16 Mb/s Token-Ring networks.

The se interface uses the standard VMS Ethernet driver to allow MultiNet to share the Ethernet device with other protocols such as LAT, LAVC, and DECnet.

s1

**Interface name:** sl0, sl1, sl2, ...sl49

**Device type:** Any VMS-supported terminal interface

<b>Parameter Prompt</b>	<b>Example Value</b>
VMS Device:	TTA0
Baud Rate:	19200
Header Compression Mode:	DISABLED
IP Address:	10.0.0.70
Point-To-Point Device IP Destination Address:	10.0.0.71
IP SubNet Mask:	255.255.255.0

	<p>The MultiNet software supports SLIP with Van Jacobson's header compression algorithm, reducing the size of the headers and increasing the bandwidth available to data. Header compression mode is determined by what both sides can support.</p>																														
rp	<p><b>Interface name:</b> rp0, rp1, rp2, ...rp49</p> <p><b>Device type:</b> Raw packet</p> <table border="0" data-bbox="235 653 966 772"> <thead> <tr> <th data-bbox="235 653 500 688">Parameter Prompt</th> <th data-bbox="748 653 966 688">Example Value</th> </tr> </thead> <tbody> <tr> <td data-bbox="235 695 391 730">IP Address:</td> <td data-bbox="776 695 902 730">10.0.0.70</td> </tr> <tr> <td data-bbox="235 737 461 772">IP SubNet Mask:</td> <td data-bbox="776 737 966 772">255.255.255.0</td> </tr> </tbody> </table> <p>The rp interface allows IP packets, normally destined for transmission, to be directed instead to a user process by way of an AF_RAWPACKET socket.</p>	Parameter Prompt	Example Value	IP Address:	10.0.0.70	IP SubNet Mask:	255.255.255.0																								
Parameter Prompt	Example Value																														
IP Address:	10.0.0.70																														
IP SubNet Mask:	255.255.255.0																														
ppp	<p><b>Interface name:</b> ppp0, ppp1, ppp2, ...ppp49</p> <p><b>Device type:</b> Any supported PPP terminal interface</p> <table border="0" data-bbox="235 1262 1409 1885"> <thead> <tr> <th data-bbox="235 1262 500 1297">Parameter Prompt</th> <th data-bbox="646 1262 857 1297">Example Value</th> </tr> </thead> <tbody> <tr> <td data-bbox="235 1304 418 1339">VMS Device:</td> <td data-bbox="690 1304 776 1339">TTA0</td> </tr> <tr> <td data-bbox="235 1346 386 1381">Baud Rate:</td> <td data-bbox="683 1346 769 1381">19200</td> </tr> <tr> <td data-bbox="235 1388 483 1423">PPP ACCM Mask:</td> <td data-bbox="683 1388 704 1423">0</td> </tr> <tr> <td data-bbox="235 1430 607 1465">PPP Authentication Method:</td> <td data-bbox="634 1430 704 1465">None</td> </tr> <tr> <td data-bbox="235 1472 597 1507">PPP Protocol Compression:</td> <td data-bbox="634 1472 704 1507">OFF</td> </tr> <tr> <td colspan="2" data-bbox="235 1514 565 1549">PPP Address and Control</td> </tr> <tr> <td data-bbox="235 1556 488 1591">Field Compression:</td> <td data-bbox="662 1556 727 1591">OFF</td> </tr> <tr> <td data-bbox="235 1598 467 1633">PPP Retry Count:</td> <td data-bbox="678 1598 1289 1633">0 (If 0, defaults to the compiled-in value of 10.)</td> </tr> <tr> <td data-bbox="235 1640 477 1675">PPP Idle Timeout:</td> <td data-bbox="672 1640 1409 1675">0 (If 0, defaults to the compiled-in value of 300 seconds.)</td> </tr> <tr> <td data-bbox="235 1682 444 1717">PPP MRU Size:</td> <td data-bbox="695 1682 716 1717">0</td> </tr> <tr> <td data-bbox="235 1724 386 1759">PPP ICMP:</td> <td data-bbox="699 1724 857 1759">ENABLED</td> </tr> <tr> <td data-bbox="235 1766 542 1801">PPP TCP Compression:</td> <td data-bbox="667 1766 732 1801">OFF</td> </tr> <tr> <td data-bbox="235 1808 634 1843">PPP Termination Retry Count:</td> <td data-bbox="646 1808 1256 1843">0 (If 0, defaults to the compiled-in value of 10.)</td> </tr> <tr> <td data-bbox="235 1850 418 1885">PPP Timeout:</td> <td data-bbox="683 1850 1409 1885">0 (If 0, defaults to the compiled-in value of 30 seconds.)</td> </tr> </tbody> </table>	Parameter Prompt	Example Value	VMS Device:	TTA0	Baud Rate:	19200	PPP ACCM Mask:	0	PPP Authentication Method:	None	PPP Protocol Compression:	OFF	PPP Address and Control		Field Compression:	OFF	PPP Retry Count:	0 (If 0, defaults to the compiled-in value of 10.)	PPP Idle Timeout:	0 (If 0, defaults to the compiled-in value of 300 seconds.)	PPP MRU Size:	0	PPP ICMP:	ENABLED	PPP TCP Compression:	OFF	PPP Termination Retry Count:	0 (If 0, defaults to the compiled-in value of 10.)	PPP Timeout:	0 (If 0, defaults to the compiled-in value of 30 seconds.)
Parameter Prompt	Example Value																														
VMS Device:	TTA0																														
Baud Rate:	19200																														
PPP ACCM Mask:	0																														
PPP Authentication Method:	None																														
PPP Protocol Compression:	OFF																														
PPP Address and Control																															
Field Compression:	OFF																														
PPP Retry Count:	0 (If 0, defaults to the compiled-in value of 10.)																														
PPP Idle Timeout:	0 (If 0, defaults to the compiled-in value of 300 seconds.)																														
PPP MRU Size:	0																														
PPP ICMP:	ENABLED																														
PPP TCP Compression:	OFF																														
PPP Termination Retry Count:	0 (If 0, defaults to the compiled-in value of 10.)																														
PPP Timeout:	0 (If 0, defaults to the compiled-in value of 30 seconds.)																														

	IP Address: 0.0.0.0 Point-To-Point Device 0.0.0.0 IP Destination Address: IP SubNet Mask: 255.255.255.0						
stf	<p><b>Interface name:</b> stf0 - only one six to four interface can exist on a system.</p> <p>Note that the CREATE command is used to create this interface.</p> <table> <thead> <tr> <th>Parameter Prompt</th> <th>Example Value</th> </tr> </thead> <tbody> <tr> <td>IPv4 address to use [NONE]:</td> <td>10.0.0.2</td> </tr> <tr> <td>Mask length [48]:</td> <td>48</td> </tr> </tbody> </table>	Parameter Prompt	Example Value	IPv4 address to use [NONE]:	10.0.0.2	Mask length [48]:	48
Parameter Prompt	Example Value						
IPv4 address to use [NONE]:	10.0.0.2						
Mask length [48]:	48						

## Displaying Interrupt Vectors

You can display the current interrupt vector used by VMS by invoking the `SYSGEN` utility, then using the `SHOW /CONFIGURATION` command, as follows:

```
$ MCR SYSGEN
SYSGEN>SHOW/CONFIGURATION
```

You can also display the maximum device configuration and the vectors currently used by MultiNet by invoking `NET-CONFIG`.

## Adding Network Interfaces with NET-CONFIG

To add an interface to the configuration:

1. Start `NET-CONFIG`:

```
$ MULTINET CONFIGURE /INTERFACES
```

2. At the `NET-CONFIG` prompt, enter:

```
NET-CONFIG>ADD interface_name
```

*interface\_name* is the name of a supported network interface device. Do not use an interface name currently in use; to modify an existing interface, see the *Modifying Network Interfaces* section. For example, to add a third shared Ethernet (se) interface to your network configuration, enter:

```
NET-CONFIG>ADD SE2
```

NET-CONFIG prompts you for interface parameter values required by the *interface\_name* interface.

3. Enter interface configuration parameter values at each NET-CONFIG prompt. For descriptions of the required parameters for your network interface, refer to *Interfaces and Parameters*.

4. When the NET-CONFIG prompt returns, verify the validity of the new interface configuration with the CHECK command.

If the CHECK command produces error messages, view the configuration parameters with the SHOW command to determine the cause of the error. To correct the error, modify the configuration as described in the *Modifying Network Interfaces* section or abandon your changes with the GET command (which reloads the configuration file) and repeat from Step 2.

6. If the CHECK command produces no error messages, quit NET-CONFIG with the EXIT command.

Your changes take effect the next time your system reboots.

```
$ MULTINET CONFIGURE
```

```
MultiNet Network Configuration Utility 5.6(nnn)  
[Reading in MAXIMUM configuration from MULTINET:MULTINET.EXE]  
[Reading in configuration from MULTINET:NETWORK_DEVICES.CONFIGURATION]  
NET-CONFIG>ADD SL0  
[Adding new configuration entry for device "sl0"]  
VMS Device [TTA0] TTA2  
Baud Rate: [UNSPECIFIED]  
Header Compression Mode: [DISABLED]  
IP Address: [NONE] 10.1.1.1  
Point-To-Point Device IP Destination Address: [NONE] 10.1.1.2  
IP SubNet Mask: [NONE]  
[sl0 (Serial Line IP): Csr=NONE, Flags=%X0]  
NET-CONFIG>EXIT  
[Writing configuration to MULTINET:NETWORK_DEVICES.CONFIGURATION]  
[Writing Startup file MULTINET:START_MULTINET.COM]  
[Changes take effect after the next VMS reboot]  
$
```



# Creating a Custom Interface Initialization Procedure

If your network requires that a device be initialized with parameters not supported by `NET-CONFIG`, you can create a custom initialization command procedure for the device. At network startup, MultiNet uses this file instead of the commands for the device in the `MULTINET:START_MULTINET.COM` file.

The device must already be part of the network configuration, and commands to its interface must already exist in the `MULTINET:START_MULTINET.COM` file. To change the device's initialization, create a command file using a text editor:

1. Create a file named `MULTINET:interface_CONFIGURE.COM`, *interface* is an interface in your configuration.
2. Copy the section of `MULTINET:START_MULTINET.COM` containing the initialization commands into the new file.
3. Edit the new file to specify the new initialization.

# Modifying Network Interfaces

To modify the configuration of an existing interface:

1. Start `NET-CONFIG`:

```
$ MULTINET CONFIGURE /INTERFACES
```

2. At the `NET-CONFIG` prompt, enter:

```
NET-CONFIG>MODIFY interface_name
```

*interface\_name* is the name of the network interface you want to modify.

For example, to modify the third shared Ethernet interface in your network configuration, enter:

```
NET-CONFIG>MODIFY SE2
```

`NET-CONFIG` prompts you for interface parameter values required by the specified interface.

3. Enter interface configuration parameter values at each of the `NET-CONFIG` prompts. For descriptions of the required parameters for your network interface, refer to the table of interface parameters earlier in this chapter.
4. When the `NET-CONFIG` prompt returns, verify the validity of the new interface configuration with the `CHECK` command.

5. If the `CHECK` command produces error messages, view the configuration parameters with the `SHOW` command to determine what is causing the error. Correct the error by repeating from Step 2, or abandon your changes with the `GET` command (which reloads the configuration file) and repeat from Step 2.

6. If the `CHECK` command produces no error messages, quit `NET-CONFIG` with the `EXIT` command.

Your changes take effect the next time your system reboots.

## Deleting Network Interfaces

You can delete network interfaces from your MultiNet configuration using `NET-CONFIG`

Use `NET-CONFIG` to delete one or all interfaces from the current configuration:

1. Start `NET-CONFIG`:

```
$ MULTINET CONFIGURE /INTERFACES
```

2. At the `NET-CONFIG` prompt, enter:

```
NET-CONFIG>DELETE interface_name
```

*interface\_name* is the name of the existing network interface you want to delete.

For example, to delete the third shared Ethernet interface in your network configuration, enter:

```
NET-CONFIG>DELETE SE2
```

3. When the `NET-CONFIG` prompt reappears, verify the validity of the new interface configuration with the `CHECK` command.

4. If the `CHECK` command produces error messages, view the configuration parameters with the `SHOW` command to determine the cause of the error. Correct the error by repeating from Step 2, or abandon your changes with the `GET` command (which reloads the configuration file) and repeat from Step 2.

5. If the `CHECK` command produces no error messages, quit `NET-CONFIG` with the `EXIT` command.

Your changes take effect the next time your system reboots.

## Enabling and Disabling Interfaces

You can disable and re-enable interfaces individually. If you disable a device, it is not configured when the network is started.

To disable a device, use the `DISABLE` command; for example:

```
NET-CONFIG>DISABLE SE0
```

To re-enable a device, use the `ENABLE` command; for example:

```
NET-CONFIG>ENABLE SE0
```

## Assigning Multiple Addresses to a Network Interface

Sometimes it is necessary to assign multiple IP addresses to a single physical interface; for example, when multiple subnets or networks are running on a single network segment.

You do this by using a pseudo device interface (`pd`). Use `NET-CONFIG` to add the device as you do other devices (such as `se` devices). This MultiNet release supports up to 500 pseudo device interfaces. Instead of specifying the OpenVMS device name, however, specify the MultiNet name for the interface (for example, `se0`). You must reboot the system after adding the pseudo device.

**CAUTION!** Careless assignment of a secondary address can cause network problems. In general, you should assign pseudo devices (`pd`) addresses on the same network or subnet as the `se` device to which the `pd` device is linked.

If the `pd` interface is not in the same IP network as its associated `se` interface, some TCP/IP packages (such as early versions of Solaris) retransmit broadcast packets for the other IP network back to the network segment from which they were transmitted. This can cause network storms.

**Note:** Some services listen to traffic on `se` interfaces only and ignore traffic on `pd` interfaces. One such service is the RIP listener in GATED.

The following example shows how to add a pseudo device:

\$ **MULTINET CONFIG**

MultiNet Network Configuration Utility 5.6  
[Reading in MAXIMUM configuration from MULTINET:MULTINET.EXE]  
[Reading in configuration from MULTINET:NETWORK\_DEVICES.CONFIGURATION]

NET-CONFIG>**SHOW**

Interface	Adapter	CSR Address	Flags/Vector
se0 (Shared VMS Ethernet/FDDI)	-NONE-	-NONE-	-NONE-
[TCP/IP: 10.1.128.20, IP-SubNet: 255.255.255.0]			
[NETWARE: A12C8000:0.0.0.0.0.0, Link Level: Ethernet]			
[VMS Device: ESA0, Link Level: Ethernet]			

|  
Official Host Name: bos.example.com  
NETWARE Host Name: BOS  
Domain Nameservers: 127.0.0.1  
Timezone: EST  
Timezone Rules: US/EASTERN  
Load EXOS \$QIO driver: TRUE  
Load UCX \$QIO driver: TRUE  
Load PWIP (Pathworks) driver: TRUE  
Nameserver Retrans Timeout: 9 (6 Retries)  
WHOIS Default Server: rs.internic.net

NET-CONFIG>**ADD PDO**

[Adding new configuration entry for device "pd0"]  
Hardware Device: [NONE] SE0  
IP Address: [NONE] **10.1.128.21**  
IP SubNet Mask: [NONE] <Return>  
Non-Standard IP Broadcast Address: [NONE] <Return>  
[pd0 (Secondary Ethernet Address): Csr=NONE, Flags=%X0]

NET-CONFIG>**SHOW**

Interface	Adapter	CSR Address	Flags/Vector
se0 (Shared VMS Ethernet/FDDI)	-NONE-	-NONE-	-NONE-
[TCP/IP: 10.1.128.20, IP-SubNet: 255.255.255.0]			
[NETWARE: A12C8000:0.0.0.0.0.0, Link Level: Ethernet]			
[VMS Device: ESA0, Link Level: Ethernet]			
pd0 (Secondary Ethernet Address)	-NONE-	-NONE-	-NONE-
[TCP/IP: 10.1.128.21]			
[Hardware-Device: se0]			

Official Host Name: bos.process.com  
NETWARE Host Name: BOS  
Domain Nameservers: 127.0.0.1  
Timezone: EST  
Timezone Rules: US/EASTERN  
Load EXOS \$QIO driver: TRUE  
Load UCX \$QIO driver: TRUE  
Load PWIP (Pathworks) driver: TRUE  
Nameserver Retrans Timeout: 9 (6 Retries)  
WHOIS Default Server: rs.internic.net

NET-CONFIG>**EXIT**

# Using Packet Filtering for Security

Packet filtering is used today in almost all (from basic to sophisticated) security firewalls. Packet filtering *firewalls* apply filtering rules to each packet received to determine whether to accept or discard it. These filtering rules specify the protocol, source and destination IP addresses, and destination ports (for TCP and UDP) for accepted or discarded packets.

You use packet filtering on routers between an internal network and one or more external networks (such as a connection to the Internet). Packet filter rules restrict what may come in through the interface connected to the external network.

Packet filtering can also be useful on hosts. For example, you can restrict the hosts that are allowed access to services. In particular, these are UDP-based services and services that the MultiNet master server does not activate, and thus cannot use incoming access restrictions.

**Note:** When you use packet filtering, each and every datagram received on the interface is filtered. This increases processing overhead depending on the size of the filter list.

Packet filtering can be an effective and useful security mechanism; however, it cannot solve all your security problems. To be effective, you must construct the filtering rules carefully.

Both IPv4 and IPv6 addresses may be filtered.

## Cautions When Creating Packet Filters

Observe the following cautions when setting up packet filtering on an interface:

- Packet filtering does not use state information. Each datagram is filtered without any knowledge of packets that preceded it. This means that for UDP-based applications, it is not possible to add a rule that says to accept replies to requests. This also affects connection-oriented protocols, such as FTP, that use two connections: one for commands and the other for data.

- Fragmented datagrams for UDP or TCP are difficult to filter, since only the first fragment has the necessary port information. MultiNet solves this problem by applying the filter rules to only the first fragment of UDP and TCP datagrams. The other fragments are accepted and processed or forwarded, but are eventually discarded because they cannot be reassembled without the first fragment. For all other IP protocols, the filter rules apply to each fragment.
- To set up secure packet filtering lists, you need detailed knowledge of IP, ICMP, TCP, UDP and applications protocols.

Suggested reading includes the protocol RFCs (listed elsewhere in the MultiNet documentation) and books such as Cheswick, William R. & Steven M. Bellovin, *Firewalls and Internet Security: Repelling the Wily Hacker*.

## Packet Filter File

Packet filtering uses a filter list to determine whether you can receive a datagram. Filter lists are in packet filter files having the .DAT extension by default. Create one of these files first and then edit the file using the formats described in the following table.

**Note:** The format of the individual filter source address and mask, and destination address and mask, has changed. In previous releases, these were specified as an IPv4 address and IPv4 mask (e.g., “192.168.0.11 255.255.255.0”). This has been changed to use addresses and masks specified in CIDR (Classless InterDomain Routing) format (e.g., “192.168.0.11/24”). This not only makes the specification of addresses and masks clearer, it also allows for the implementation of IPv6 addresses which are substantially longer than IPv4 addresses, leading to potential problems with long filter file lines. The `FILTER_CONVERT` utility is provided to change existing filter files from the old format to the new format.

Field...	With valid values...	Means...
action protocol	saddr [sport]	daddr [dport [doption]]
action	permit deny drop	permit allows the datagram; deny denies the datagram and sends the ICMP;

		drop denies the datagram without sending an ICMP destination unreachable message to the sender.
protocol	ip <i>ip-number</i> tcp udp icmp	Protocol to check: the values indicated or the numeric IP protocol number. The value <i>ip</i> matches any IP protocol.
saddr	<b>Example:</b> 192.168.123.123/24	Source IP address to check in CIDR format. This may be in ipv4 or ipv6 format.
sport	operator <i>operand</i> lt <i>port</i> le eq ge gt ne	Optional source port specification to check (for TCP and UDP entries only). Consists of an operator, space, and port name or number. If port name, must be defined in the MULTINET:HOSTS.SERVICES file. If omitted, any source port is valid. Example: gt 1023
daddr	<b>Example:</b> 192.168.123.123/240	Destination IP address to check in CIDR format. This may be in ipv4 or ipv6 format, but must be of the same address family as saddr
dport	operator <i>operand</i> lt <i>port</i> le <i>icmp-msg-type</i> eq ge gt ne	Optional destination port (for TCP and UDP entries) or ICMP message type specification consisting of an operator, space, and operand. If a port name, must be in the MULTINET:HOSTS.SERVICES file. If <i>icmp-msg-type</i> , use:  0-Echo Reply 3-Destination Unreachable 4-Source Quench 5-Redirect 8-Echo 11-Time Exceeded 12-Parameter Problem 13-Timestamp 14-Timestamp Reply 15-Information Request 16-Information Reply

doption	established	Matches only established connections (TCP segments with ACK or RST bits set).
start time		<p>VMS-format absolute or delta date/time.</p> <p>If specified, the filter takes effect starting at the time specified.</p> <p>By default, a filter takes effect when loaded by MultiNet if the start time is not defined.</p>
end time		<p>Absolute or delta VMS-format date/time.</p> <p>If specified, the filter is ignored after the time specified is reached if an absolute time is specified, or after the time calculated by adding the end time to the start time if a delta time is specified.</p> <p>If no start time was specified, the delta time is added to the current time.</p> <p>If the end time for a non-repeating filter has already passed when the filter definition is parsed, the filter is not loaded and the user is informed of that fact.</p>
repeat		If Y and start/end times are specified, this filter repeats until it is removed. Both a start and end time must be specified in order to specify a repeating filter.
log		Logs events from this filter.



Each entry specifies a packet filtering condition for a particular protocol type, source or destination address and mask, and destination port and mask specification, with certain additional options. The system looks at each condition in sequence, looks for a match, and takes a permit (accept) or deny (reject) action. The system stops testing conditions after the first match. This means that the order of the entries in the file is important; if the file lists a subsequent condition for an address, the system ignores it.

An implicit deny terminates the list of entries in the packet filter file. This means that if no condition matches, the system rejects the datagram. To use packet filtering:

1. Create address list entries in the packet filter file.
2. Apply the list to interfaces on your system by using packet filtering commands.

To create a packet filter file, edit a file and add address list entries in the format described.

Any number of spaces or tabs can separate each entity. Lines beginning with an exclamation point (!) are comment lines. You can use the dash (-) continuation character at the end of a line that needs to continue onto the next.

To apply the list to a particular network interface or interfaces on your system, use the `MULTINET SET/INTERFACE/FILTER` command, as described in *MultiNet Administrator's Reference*.

## Configuration Recommendations

Constructing an address filter list requires care in that you want to allow only the packets you need. Here are some recommendations in setting up an address filter list for an interface:

- Add an entry to prevent IP "spoofing"-having an external host send a datagram as if it came from a local machine. For a router, this makes sense because no datagram received from an external network should ever have a local source address. Add the following entry to the filter list for the external interface:

```
deny ip local-network
```

- Be careful with services that use "unprotected" port numbers (greater than 1024). Some examples are NFS (port 2049) and X Windows (port 6000 and higher). Explicitly denying these services is a good idea:

```
deny udp 0/0 0/0 eq 2049
deny tcp 0/0 0/0 eq 2049
deny tcp 0/0 0/0 eq 6000
deny tcp 0/0 0/0 eq 6001
```

- Prevent broadcast and loopback packets from entering your network. It is best to restrict the broadcast (the first two of the following entries) to an external interface; apply the loopback restriction (the last entry) to any interface:

```
drop ip 0.0.0.0/32
drop ip 255.255.255.255/32
drop ip 127.0.0.0/8
```

- Guard against datagrams from invalid source addresses when connected to the Internet (provided you are not using these network numbers for internal-only traffic purposes). Add the following to the filter list for the external interface:

```
drop ip 10.0.0.0/8
drop ip 172.16.0.0/12
drop ip 192.168.0.0/16
```

- You generally need to allow domain name (DNS) requests using:

```
permit udp 0/32 eq 53 0 0
```

Whether to allow TCP DNS traffic (usually used for zone transfers) is also something to consider. To disallow TCP DNS traffic, add:

```
deny tcp 0/32 eq 53 0 0
```

- You should not be concerned with what services local users use in the external world. You would want to add:

```
permit tcp 0/32 0/32 gt 1023 established
```

This allows all TCP datagrams in to ports greater than 1023 that have either the ACK or RST bits set in the TCP flags. Connection establishment requests have just the SYN bit set, so they are not allowed by this entry.

You might want to drop the established option if you want to allow incoming connections to unprotected ports. This would allow use of the FTP PASV capability.

- You may offer services to the external world such as a World Wide Web or anonymous FTP server. Add the following entries:

```
permit tcp 0/32 web-server-address/32 eq 80
permit tcp 0/32 ftp-server-address/32 eq 21
```

If you have several hosts for each service, add an entry for each.

**Note:** For the FTP Server, the data connections are normally outgoing and thus the earlier `permit tcp 0 0 0 0 gt 1023 established` configuration works to allow these. However, if users switch to PASV mode, the connections will be incoming (to unprotected

port numbers) and therefore the `permit tcp 0 0 0 0 gt 1023` configuration (without the established option) might be more effective.

- Allow all ICMPs except ICMP redirects:

```
deny icmp 0/32 0/32 eq 5
permit icmp
```

This is useful for informing hosts about problems. But it can open up denial of service attacks, especially if hosts are not careful about the ICMP redirects they accept. That is why discarding them is recommended.

- Watch the order of the entries in the table carefully.

```
permit tcp 0/32 0/32 gt 1023
deny tcp 0/32 0/32 eq 2049
```

This entry would not work since the permit entry allows the datagram and processing stops as soon as a match is found. MultiNet processes the entries in the order in which you specify them.

- Remember that an implicit "deny everything" is added to the end of the filtering list. This means that to permit a datagram, you need to have a permit entry in the list.
- Once you applied your filter list, test it first. Get an account on a host on an outside network that you can use to connect to your local hosts. Check that you are not allowing any access you do not want, and that you are allowing access that you do want. If something is not right, modify the filter list, reload it, and retest.

While packet filtering is very useful, it is by no means the only step you should take to secure your network. You must take special care to secure the system to assure that it cannot be compromised. One way to do this is to greatly limit the services it offers.

## Filtering by Time

Filters may be set to be active only during a specified time period. These filters may be specified as a one-time filter or as a filter that repeats. For example, a filter may be set up to filter all traffic from a specific address during the hours of 5am to 5pm each day, or a filter may be specified that filters traffic starting from the time the filter is loaded and for the next 3 hours.

Time-based filtering is done by specifying a start time, an end time, or both start and end times for a filter in the filter definition file. For repeating filters, both start and end times must be specified. Note

that all time values for start and end times must be specified in VMS absolute or delta time format. For example, the following are all valid:

“29-DEC-2003”	would be interpreted as "29-DEC-2003 <current time>"
“29-DEC-2003 18:03”	would be interpreted as "29-DEC-2003 18:03:00.00"
18:03:00	would be interpreted as "<current date> 18:03:00.00"
“1 15:00”	would be interpreted as a delta time of 1 day, 15 hours

Note that if an absolute time is specified that contains both a date and time (example 2 above), it **MUST** be enclosed by double quotes. For example:

```
deny icmp 0 0 eq 5 start 17:00:00 end "29-Sep-2003 6:00:00"
```

Given the following filter file:

```
deny tcp 15.1.94.2/32 2.22.2.5 255/32 start 15:20 end 18:30 repeat
deny tcp 1.1.94.2/32 207.225.29.51/32 end "1-JAN-2004 18:30"
deny tcp 195.101.94.209/32 207.225.29.51/32 start 18:00 end "1 00:30" repeat
deny tcp 195.101.94.209/32 207.225.29.51/32
deny tcp 195.101.94.209/32 207.225.29.51/32 start 17:00 end 18:30
deny tcp 15.1.94.2/32 2.22.2.5/32 start "2 00:00" end 3 00:00"
```

Line 1 will filter from 15:20 to 18:30 each day.

Line 2 will filter from the time the filter is loaded through 18:30 on January 1, 2004 with no logging. After that time, if the filters are reloaded, this filter will not be loaded.

Line 3 will filter from 18:00 to 19:30 each day.

Line 4 has no time limits on it.

Line 5 will log from 17:00 through 18:30 today.

Line 6 will filter starting 2 days from the time the filter is loaded, through 3 days after that.

## Filter Logging

Filter "hits" may be logged, either to OPCOM or to a file defined by the user. Logging is enabled on a filter-by-filter basis, by using the "log" keyword on the end of a filter definition line. For example:

```
deny tcp 192.10.9.209/32 207.225.29.51/32 log
```

Logging for the interface is controlled via the MULTINET SET/INTERFACE command. The actual logging is performed by the MULTINET\_FLOG process, which is started the first time a MULTINET SET/INTERFACE /LOG command is issued (a single MULTINET\_FLOG process handles logging for all interfaces defined on the system).

The MULTINET SET/INTERFACE command switches used to support logging are

Qualifier	Values	Default	Description
/[NO] LOGGING	OPCOM or a valid filename	None	Used to turn logging on or off. Filter events may be logged to OPCOM or to a specified file. Only those events with the LOG qualifier in their definition are affected by this qualifier.
/FORMAT	COMMA or NORMAL	NORMAL	If NORMAL, then the normal formatting as seen by MU SHOW/INTERFACE/FILTER will be used. If COMMA, then a comma-delimited file will be created that can be, for example, loaded into a spreadsheet.
/INTERVAL	Number of seconds, between 5 and 2 <sup>31</sup>	5 seconds	Reporting interval. The minimum reporting interval is 5 seconds, so that a flood of filter events doesn't drag the system down. When reporting events, a count of missed events will be included for each event where the event couldn't be reported before the next event occurred.

When filter logging is enabled, the MULTINET\_FLOG process will be started. This process checks each interface at the interval defined by the /INTERVAL qualifier for the MULTINET SET/INTERFACE command. As unlogged filter hits are found, it will log them to OPCOM or to a file, according on the parameters set by the /LOG and /FORMAT qualifiers for the MULTINET SET/INTERFACE command.

When logging to OPCOM, only NORMAL formatting is allowed. An OPCOM message, formatted as the filter output from MULTINET SHOW/INTERFACE /FILTER, will be displayed for each filter with unlogged hits on it.

When logging to a file, the output will be identical to that of the filter displays from MULTINET SHOW/INTERFACE /FILTER command, if /FORMAT=NORMAL is specified. If /FORMAT=COMMA is specified, the data will be recorded as comma-delimited fields, one line per filter, to the file. The first line of this file will contain the field names (comma-delimited) to aid in interpreting the contents of the file.

Examples:

```
$ MULTINET SET /INTERFACE SE0 /LOG=OPCOM/INTERVAL=10
```

enables logging to OPCOM, with a reporting interval of 10 seconds.

```
$ MULTINET SET /INTERFACE SE0 /LOG=FOO.DAT/FORMAT=COMMA
```

enables logging to the file `FOO.DAT` in comma-delimited format, and a reporting interval of 5 seconds (the default).

```
$ MULTINET SET /INTERFACE SE0 /NOLOG
```

This disables all logging for the interface, closing all open log files.

## Setting the Filter List at Startup

When you start MultiNet, the `START_MULTINET` procedure looks for a `MULTINET:FILTER-interface.DAT` file for each interface it starts. If the file exists, `START_MULTINET` issues the following command to set the filter list for the interface:

```
$ MULTINET SET/INTERFACE interface /FILTER=MULTINET:FILTER-line-id.DAT
```

You can also add the necessary MultiNet commands to the `MULTINET:LOCAL_ROUTES.COM` file.

If you want to know if filtering is enabled and what the settings are, use the `MULTINET SHOW /INTERFACE/FILTER SE0` command.

MultiNet also supports the use of a `LOCAL_INITIALIZATION` command procedure during startup. As with the `MULTINET:LOCAL_ROUTES.COM` file, you can put the necessary MultiNet filter commands in the `MULTINET:LOCAL_INITIALIZATION.COM` file to have them executed as MultiNet starts.

## Converting an Old-Format Filter File

The `FILTER_CONVERT` utility is provided to convert from the old-format filter file (one which uses separate address/mask fields) to the new-format filter file (one which uses CIDR format address specification). To use this:

```
$ FILTER_CONVERT := $MULTINET:FILTER_CONVERT
$ FILTER_CONVERT infile outfile
```

When a filter file has been converted, the resulting output file should be checked for correctness prior to using it.

# Configuring Transport over Serial Lines with SLIP and PPP

MultiNet supports remote IP transport over serial lines with SLIP (Serial Line IP) or PPP (Point-to-Point Protocol).

## Understanding SLIP and PPP

Both SLIP and PPP use a simple framing protocol to transfer datagrams over a terminal line. Both require an RS232-C serial line, or one that looks like an asynchronous line to VMS.

MultiNet SLIP interfaces (identified by the name `sl`) and PPP interfaces (identified by the name `ppp`) can send and receive packets over any asynchronous terminal line (OpenVMS device of types `TTcn` or `TXcn`) connected to other systems that support the SLIP and PPP protocols, respectively.

As a result, MultiNet systems can communicate asynchronously with other MultiNet systems (or UNIX and other systems) that support SLIP or PPP.

With SLIP or PPP, users on one system can connect with another system using modems over telephone lines or over hardwired connections. To use SLIP or PPP over modems, the modems must be 8-bit transparent so all 256 ASCII codes can be sent and received.

If the remote system is configured as a gateway to a network, local users can also reach other systems on that network. MultiNet supports SLIP and PPP running over any VMS-supported terminal multiplexer. MultiNet does not support SLIP or PPP over LAT.

You must know the IP address of the serial interface on every remote host with which you establish serial communications. If you configure MultiNet with PPP interfaces for multiple remote hosts, the remote hosts can obtain their IP addresses when they connect. Similarly, you can configure a PPP interface on MultiNet without knowing your own IP address, and obtain it when you connect to a remote system.

The two methods of connecting hosts via PPP or SLIP are *dynamic* and *static*. The following sections explain these methods.

## Dynamic Interfaces-Defined

The usual SLIP or PPP configuration consists of two systems connected by serial line only when needed. For these situations, configure a *dynamic* SLIP or PPP interface. Dynamic interfaces are not associated with a specific OpenVMS device until the remote host connects to a device.

For a dynamic interface, you do not specify an OpenVMS device name when configuring the interface. When MultiNet starts, new dynamic interfaces are available for serial communication; however, an administrator must attach the interfaces to VMS devices.

## Static Interfaces-Defined

Large organizations often use SLIP and PPP to connect separate LANs into a single wide area network (WAN) with dedicated serial lines. The host at each end of the serial connection is always the same, and no other hosts are allowed to connect to either serial device. In these situations, configure a *static* SLIP or PPP interface. Static interfaces are attached to a specific OpenVMS device, which prevents the serial device from being used for any other purpose.

For a static interface, you specify an OpenVMS device name when configuring the interface.

As soon as you connect two static interfaces via modem or hardwired connection, they can communicate over the chosen serial protocol; no user authentication is required.

**Note:** Because IP connectivity is established as soon as the two serial interfaces connect, do not configure static interfaces for public dial-in access.

## Configuring Static SLIP Interfaces

To configure a static SLIP interface:

1. Use `NET-CONFIG` to add the SLIP interface as described in the *Adding Network Interfaces* section. The SLIP Configuration Parameters section describes the interface parameters you must define for basic SLIP operation. Be sure to specify an OpenVMS device name.
2. If desired, create a custom startup command procedure for the new interface. For details, see the *Configuring Permanent SLIP and PPP Interfaces* section.



3. Reboot your system.

When MultiNet starts, you can connect your serial device to a remote system.

## Configuring Dynamic SLIP Interfaces

To configure a dynamic SLIP interface:

1. Use `NET-CONFIG` to add the SLIP interface as described in the *Adding Network Interfaces* section. The *SLIP Configuration Parameters* section describes the interface parameters you must define for basic SLIP operation. Do not specify an OpenVMS device name.
2. If desired, create a custom startup command procedure for the new interface as described in the *Configuring Permanent SLIP and PPP Interfaces* section.
3. Reboot your system.
4. The new dynamic interfaces are created when MultiNet starts, but they are not yet connected to a VMS device. See the *Attaching Dynamic SLIP or PPP Interfaces to VMS Devices* section for instructions.

## SLIP Configuration Parameters

The below table lists the configuration parameters for configuring static and dynamic SLIP interfaces.

Parameter	Description
SLIP interface name	Determines the interface name. Must be of the form <code>s1n</code> , <ul style="list-style-type: none"><li>• <code>n</code> is a positive integer.</li><li>• <code>s10</code> is a suitable interface name for the first SLIP interface.</li><li>• <code>s11</code> is a suitable interface name for the second one.</li></ul>
OpenVMS device name	For a <i>static</i> SLIP interface (one that uses a hardwired terminal line or dedicated modem and telephone line), specify a device name.  For a <i>dynamic</i> SLIP interface (one to which you assign a device name when the connection is made, for example, by modem dial-up), do not specify a name.

	<p>When a modem hangs up on a dynamic SLIP interface, each end of the SLIP interface automatically reverts to a normal terminal line.</p> <p>Use a value of "none" to override any specified device name. Doing so might be useful to make a previously configured interface dynamic.</p>
Baud rate	Data transfer baud rate (110, 300, 1200, 2400, 4800, 9600, 19200, or UNSPECIFIED) of the SLIP interface.
SLIP compression mode	<p>MultiNet SLIP supports the Van Jacobson header compression algorithm to reduce the bandwidth required for the TCP and IP headers. If both sides of a SLIP interface support compression, turnaround improves significantly.</p> <p>Compression modes are:</p> <ul style="list-style-type: none"> <li>• ENABLED - Headers should always be compressed.</li> <li>• DISABLED - Headers should never be compressed.</li> <li>• NEGOTIATED - Headers should not be compressed until a compressed header is received from the other side. Negotiated compression is useful on dialup gateways that do not know if the other side of SLIP interfaces support compression.</li> </ul> <p><b>Note:</b> Compression must be enabled (not just negotiated) on at least one side of a SLIP interface. Disable SLIP compression for compatibility with SLIP interfaces, such as previous releases of MultiNet that do not support it.</p>
IP address	IP address associated with the SLIP interface on the local system.

Point-to-point IP destination address	IP address associated with the SLIP interface on the target system.
---------------------------------------	---

## Configuring Static PPP Interfaces

To define a static PPP interface:

1. Use `NET-CONFIG` to add a PPP interface to your network configuration as described in the *Adding Network Interface* section. The *PPP Configuration Parameters* section describes the interface parameters you must define for basic PPP operation. Be sure to define an OpenVMS device for the interface.
2. If desired, create a custom startup command procedure for the new interface. For details, see the *Configuring Permanent SLIP and PPP Interfaces* section.
3. Reboot your system.

After rebooting, the interfaces are attached to VMS terminal lines.

## Configuring Dynamic PPP Interfaces

To define a dynamic PPP interface:

1. Use `NET-CONFIG` to add a PPP interface to your network configuration as described in the *Adding Network Interfaces* section. *PPP Configuration Parameters* describes the interface parameters you must define for basic PPP operation. Do not define an OpenVMS device for the interface.
2. Enable virtual terminal (VTA) devices for your dynamic PPP interfaces. For information on configuring virtual terminals, see your OpenVMS system management documentation.

**Note:** Dynamic PPP interfaces require `CMKRNL`, `LOG_IO`, and `SYSPRV` privileges. Because these privileges are potentially dangerous, login name and password information should be safeguarded carefully.

3. If desired, create a custom startup command procedure for the new interface. For details, see the *Configuring Permanent SLIP and PPP Interfaces* section.

4. Reboot your system.

The dynamic interfaces are created when MultiNet starts, but they are not associated with a VMS terminal line. The interfaces must now be attached to terminal lines; see the *Attaching Dynamic SLIP or PPP Interfaces to VMS Devices* section.

## PPP Configuration Parameters

The table below lists the configuration parameters for static and dynamic PPP interfaces.

Parameter	Description
PPP interface name	Determines the interface name. Must be of the form <code>pppn</code> , <ul style="list-style-type: none"><li>• <code>n</code> is a positive integer.</li><li>• <code>ppp0</code> is a suitable interface name for a first PPP interface.</li></ul>
OpenVMS device name	Dedicates the device name to a <i>static</i> PPP interface (one that uses a hardwired terminal line or dedicated modem and telephone line).  To configure a <i>dynamic</i> PPP interface (one to which you assign a device name when a connection is made—for example, by modem dial-up), do not specify a name. When a modem hangs up on a dynamic PPP interface, each end of the PPP connection automatically reverts to a normal terminal line.  Use a value of <code>none</code> to override any specified device name. Doing so might be useful to make a previously configured interface dynamic.
Baud rate	Determines the data transfer baud rate (110, 300, 1200, 2400, 4800, 9600, 19200, or UNSPECIFIED) of the PPP interface.

<p>Asynchronous Control Character Map (ACCM) Mask</p>	<p>A 32-bit mask that indicates the set of ASCII control characters to be mapped into two-character sequences for transparent transmission over the line. Default: %x00000000.</p> <p>The map is sent the most significant 8 bits first. Each numbered bit corresponds to the ASCII control character of the same value. If the bit is cleared to zero, the corresponding ASCII control character need not be mapped. If the bit is set to one, the corresponding ASCII control character must remain mapped.</p> <p>For example, if bit 19 is set to zero, the ASCII control character 19 (DC3, Control-S) can be sent in the clear.</p> <p>The least significant bit of the least significant 8 bits (the final 8 bits transmitted) is numbered bit 0, and corresponds to the ASCII control character "NUL."</p>
<p>Protocol Compression</p>	<p>When ON, PPP negotiates with the peer to use one byte instead of two for the Protocol fields to improve transmission efficiency on low-speed lines. Default: OFF.</p>
<p>Address and Control Field Compression (ACFC)</p>	<p>When ON, PPP eliminates the address and control fields when they are identical over a series of frames. Default: OFF.</p>
<p>Retry Count</p>	<p>Determines the number of attempts PPP makes to configure a connection with "Configure-Request" packets. Default: 10.</p>
<p>Idle Timeout</p>	<p>Determines how long (in seconds) the connection must remain idle before PPP attempts to close the connection with "Terminate-Request" packets. Default: 300.</p>
<p>Maximum Receive Unit (MRU) Size</p>	<p>Determines the maximum number of 8-bit bytes for the PPP Information field, including padding, but not including the Protocol field. Because opposite ends</p>

	of a PPP connection may have different MRU values, PPP negotiates a suitable MRU for both systems. Default: 1500.
ICMP	When ENABLED, PPP allows ICMP packets over the PPP connection. Administrators may want to disable ICMP packets if they are concerned with "service attacks" from dial-up connections. Default: ENABLED.
TCP Header Compression	When ENABLED, requests the IPCP driver to employ Van Jacobson TCP header compression to improve performance. Default: DISABLED.
Termination Retry Count	Determines the number of attempts PPP makes to terminate a connection with "Terminate-Request" packets. Default: 10.
Timeout	Determines the time (in seconds) between successive Configure-Request or Terminate-Request packets. Default: 30.
IP Address	The IP address of the local PPP interface in dotted-decimal format. You may also specify 0.0.0.0 (the default) to indicate that the local IP address will be specified by the remote peer when the serial connection is established.
Point-To-Point Device IP Destination Address	The IP address of the peer PPP interface in dotted-decimal format. Default: 0.0.0.0.
SubNet Mask	The subnet mask of the local PPP interface in dotted-decimal format. The default depends on the local PPP interface IP address. For example, a class A address results in a default subnet mask of 255.0.0.0.

## Configuring Permanent SLIP and PPP Interfaces

When you configure an interface, the following line is added to your `START_MULTINET.COM` file to initialize the device:

```
$ SET TERM/PERM/NOTYPE_AHEAD/NOAUTOBAUD/SPEED=config_speed dev_name
```

- `config_speed` is the baud rate for transmitting and receiving data.
- `dev_name` specifies a SLIP or PPP interface such as `TTA1 :.`

This setting is used for permanent interfaces to prevent LOGINOUT from gaining control of the device because of extraneous noise on the line.

If you want to override this behavior, create a custom command procedure, `MULTINET:dev_name_CONFIGURE.COM`, for that interface. If you have an IP address assigned to the device, your custom command procedure is invoked at startup instead of the command line described previously.

## Attaching Dynamic SLIP or PPP Interfaces to VMS Devices

After a remote host connects to a MultiNet system over a serial line, the remote system administrator must log into the MultiNet system and attach the appropriate MultiNet dynamic interface to the VMS terminal line to which the remote host is connected.

For example, if your system provides serial access via two modems, but you need to provide access to three or more hosts, configure a dynamic interface for each host you plan to accommodate. Then make sure the administrator on each remote host knows the name of the serial interface you have configured and the commands to execute to attach SLIP or PPP interfaces to the appropriate terminal lines.

To convert a terminal line into a SLIP or PPP interface:

1. Log into an account with `CMKRNL`, `LOG_IO`, and `SYSRV` privileges.
2. Determine the name of the serial interface corresponding to the remote host. If the administrator knows the remote IP address or host name, the corresponding serial interface name can be determined from the `MULTINET SHOW` command output. For example:

```
$ MULTINET SHOW /STATISTICS
MultiNet Network Interface statistics:
Name  Mtu   Network Address          Ipkts   Ierrs  Opkts   Oerrs  Collis
----  ---  -
se0   1500  ABC-NET FOO.BAR.COM      120360  0      143384  1      4
sl0   296   ABC-NET FOO.BAR.COM       0       0       1       0      0
lo0   1536  LOOPBACK-NET LOCALHOST    917     0       917     0      0
$ MULTINET SHOW /INTERFACE
_Network Device: SL0
Device sl0: flags=71<UP, POINTOPOINT, NOTRAILERS, RUNNING>
IP Address = 10.41.228.78
IP Sub-Net Mask = 255.255.255.192
IP Point to Point Destination = 10.41.228.80
```

3. Attach the serial interface to the VMS device with the following DCL command:

```
$ MULTINET SET /INTERFACE interface_name /DYNAMIC /VMS_DEVICE
/LINK LEVEL=protocol
```

*protocol* is SLIP or PPP, according to the type of serial interface.

**Note:** If the remote host is also a MultiNet system, the remote administrator must also attach the remote serial interface to the remote VMS device.

The following example illustrates how to establish connectivity between two MultiNet systems over dynamic SLIP interfaces. The first `MULTINET SET /INTERFACE` command converts the remote host's dial-in port into a SLIP interface. After the user types the control-backslash `Ctrl/\` escape key sequence to return to a local DCL command line, the second `MULTINET SET /INTERFACE` command converts the local terminal line into a SLIP interface. The `MULTINET PING` command confirms IP connectivity.

```
WHARFIN$ ALLOCATE TTA1:
%DCL-I-ALLOC, WHARFIN$TTA1: allocated
WHARFIN$ SET TERMINAL/SPEED=2400 TTA1:
WHARFIN$ SET HOST/DTE TTA1:
%REM-I-TOEXIT, connection established, type ^\ to exit
ATDT1415555-1212
RRING
CONNECT 2400
BIGBOOTE SLIP Gateway - VAX/VMS V5.5-2
Username: SYSTEM
Password: *****
Welcome to the BIGBOOTE SLIP Gateway
Last interactive login on Monday, 28-FEB-2020 14:47
Last non-interactive login on Monday, 28-FEB-2020 13:16
BIGBOOTE$ MULTINET SET/INTERFACE SL1/DYNAMIC/LINK LEVEL=SLIP/VMS_DEVICE
The line you are logged in over is now a SLIP line.

[You now type the Ctrl/\escape sequence to return to WHARFIN.]

%REM-S-END, control returned to node WHARFIN::
WHARFIN$ MULTINET SET/INTERFACE SL1/DYNAMIC/LINK LEVEL=SLIP/VMS_DEVICE=TTA1:
WHARFIN$ MULTINET PING 192.0.0.3
PING 192.0.0.3 (192.0.0.3): 56 data bytes
64 bytes from 192.0.0.3: icmp_seq=0 time=860 ms
64 bytes from 192.0.0.3: icmp_seq=1 time=860 ms
64 bytes from 192.0.0.3: icmp_seq=2 time=860 ms
Ctrl+C
----192.0.0.3 PING Statistics----
4 packets transmitted, 3 packets received, 25% packet loss
```



```
round-trip (ms)  min/avg/max = 860/860/860
WHARFIN$
```

## Shutting Down a PPP or SLIP Interface

To bring down an interface, issue the following command:

```
$ MULTINET SET /INTERFACE interface_name /LINK_LEVEL=protocol
/VMS_DEVICE=device /DOWN
```

## Modifying Global Parameters

MultiNet maintains a set of global parameters that affect the behavior of all network interfaces. For example, your system's default route (see Chapter 13) is specified as a global parameter and affects how all network interfaces direct data packets over the network.

You can configure global parameters with NET-CONFIG, using the `SET parameter_name` command. See the *MultiNet Administrator's Reference* for a complete list of SET commands.

You can modify all global parameters without rebooting your system with the following exceptions:

```
LOAD-EXOS-DRIVER
LOAD-UCX-DRIVER
WINS-COMPATIBILITY
```

Changes to these parameters take effect after you reboot the system.

The following subsections describe how to modify global parameters to perform the following tasks:

- Configuring DECwindows support (see the *Using the HP TCP/IP Transport Over UCX* section)
- Configuring the cluster alias feature, which permits another VMScluster node to continue to provide some connectionless network services if the primary node that provides those services fails (see the *Configuring VMScluster Aliasing* section)
- Ensuring PATHWORKS 5.0 support is enabled (see the *Ensuring PATHWORKS Support is Enabled* section)

# Using the HP TCP/IP Transport Over UCX

You can configure the DECwindows server and X clients to use the HP TCP/IP Transport over MultiNet's UCX driver. MultiNet supports DECwindows over TCP/IP under VMS V5.5-2 and later versions by emulating the HP TCP/IP Services for the VMS \$QIO interface.

The MultiNet UCX driver is enabled by default.

1. Edit your system startup command procedure to invoke MultiNet before starting DECwindows.
2. Reboot your system to start MultiNet with the UCX \$QIO driver loaded.
3. To create a display, issue the following command:

```
$ SET DISPLAY/CREATE/NODE=ip-node-name /TRANSPORT=TCPIP
```

For complete information on the SET DISPLAY command and running remote DECwindows applications, see the *VMS DECwindows User's Guide*.

Each user must enable TCP/IP access on a host-by-host basis using the DECwindows Session Manager Customize Security menu so they can run applications over the TCP/IP transport.

From that menu, specify:

- TCP/IP as the Transport
- The remote host's Internet host name as the Node
- A question mark (?) for the Username

The DECwindows chapter of the *MultiNet User's Guide* contains information about running DECwindows applications over MultiNet.

## Configuring VMSccluster Aliasing

If you have the cluster alias feature configured on more than one node in a VMSccluster, the nodes negotiate among themselves so that only one node at a time answers requests to the cluster alias.

If the node serving the cluster alias fails and more than one node has the cluster alias configured, the service provided by the failed node is provided by another node in the cluster. Without the cluster alias feature, the services provided by the failed node are not available.

To use cluster aliasing, you provide a list of addresses to which each node will answer. If the node currently serving the cluster alias fails, one of the other nodes takes over the connectionless services for that address.

This feature lets you specify one or more nodes in the cluster as having an additional IP address, so only one of the nodes will use that additional IP address at any one time.

Enable cluster aliases with the `NET-CONFIG SET IP-CLUSTER-ALIASES` command. Specify the extra IP addresses for which this node should also answer requests. Disable cluster aliases by invoking the command without addresses. You can change the value of `IP-CLUSTER-ALIASES` without rebooting by also defining or redefining the system-wide logical name `MULTINET_IP_CLUSTER_ALIASES` and restarting the `MULTINET_SERVER` process.

Since this service is disabled by default, you must enable it in this way:

```
$ MULTINET CONFIGURE /SERVER  
SERVER-CONFIG>ENABLE CLUSTERALIAS  
SERVER-CONFIG>EXIT
```

Now you can enable cluster failover as shown in the following example:

```
$ MULTINET CONFIGURE  
MultiNet Network Configuration Utility 5.6(nnn)  
[Reading in MAXIMUM configuration from MULTINET:MULTINET.EXE]  
[Reading in configuration from MULTINET:NETWORK_DEVICES.CONFIGURATION]  
NET-CONFIG>SET IP-CLUSTER-ALIASES 10.1.1.2  
NET-CONFIG>EXIT  
[Writing configuration to MULTINET:NETWORK_DEVICES.CONFIGURATION]  
[Writing Startup file MULTINET:START_MULTINET.COM]  
[Changes take effect after the next VMS reboot]  
$ DEFINE /SYSTEM /EXECUTIVE MULTINET_IP_CLUSTER_ALIASES "10.1.1.2"  
$ @MULTINET:START_SERVER
```

## Ensuring PATHWORKS Support is Enabled

By default, MultiNet is configured to support Pathworks 5.0 running concurrently. To ensure this support is enabled:

1. Verify the presence of the PWIP (Pathworks over IP) device:

```
$ SHOW DEV PWIP
```

2. Invoke `NET-CONFIG` and enter the `SHOW` command. Check the "Load PWIP (Pathworks) driver:" line.

3. If MultiNet is not configured to load the PWIP driver, enter the `SET LOAD-PWIP-DRIVER` command.
4. Save the configuration to ensure it is loaded the next time your system reboots.

## Multicast Support

To enable multicast packet reception under OpenVMS V5.5-2, create a custom `SE0_CONFIGURE.COM` file that adds the `/MULTICAST=ALL` qualifier to the `MULTINET SET /INTERFACE` command line in that command procedure.

Multicast reception is enabled automatically in OpenVMS VAX V6.0 or later and in all versions of OpenVMS Alpha. The file `MULTINET_ROOT:[MULTINET.EXAMPLES]SE0_CONFIGURE.COM` is a template for such command procedures.

## Enabling and Disabling MTU Discovery

Maximum Transmission Unit (MTU) discovery determines the maximum size of a TCP packet that can be sent through the network between two hosts. Performance improves when the largest, most efficient packet size possible with the hardware at each hop is enabled. RFC-1191 describes this feature, which is enabled by default.

When MTU discovery becomes active for a remote host, it places a host route in the routing table with the MTU set to the appropriate size. This feature is potentially useful for tracing unusual routes.

MTU discovery sets the Don't Fragment (DF) bit in IP packets. It is difficult to predict how routers from different vendors will handle the DF bit; some handle it correctly, some do not, some work until they need to fragment a packet, and some simply drop the packet. If you suspect a routing problem is affecting communications, disable MTU discovery by issuing the following command:

```
$ MULTINET SET/KERNEL TCP_PMTU 0
```

To enable it again, issue this command:

```
$ MULTINET SET/KERNEL TCP_PMTU 1
```

Both of these commands take effect immediately.

# Manipulating the ARP Table

The Address Resolution Protocol (ARP) dynamically maps addresses between Internet and Ethernet. ARP is used by all MultiNet Ethernet interface drivers and HP Computer FDDI drivers.

ARP caches Internet-Ethernet address mappings. When an interface requests a mapping for an address not in the cache, ARP queues the message requiring the mapping and broadcasts an ARP request on the associated network requesting the address mapping.

If a response is provided, the new mapping is cached in the ARP table and any pending messages are transmitted. ARP queues no more than one packet while waiting for a mapping request to be responded to; only the most recently "transmitted" packet is kept.

To enable communications with systems that do not use ARP, the `MULTINET SET /ARP` utility allows you to add and delete entries in the Internet-to-Ethernet tables.

**CAUTION:** Adding or modifying entries in the ARP table can seriously affect TCP/IP communications. Do not create or modify ARP table entries unless you are sure of their effects on your network.

The `SET/ARP` qualifiers are:

Qualifier	Description
<code>/ADD</code>	Adds a specified host-to-Ethernet address translation to the ARP tables
<code>/DELETE</code>	Deletes a specified host-to-Ethernet address translation from the ARP tables
<code>/FLUSH</code>	Flushes temporary entries in the ARP tables
<code>/PERMANENT</code>	Used with <code>/ADD</code> or <code>/FLUSH</code> to specify whether an entry is added or flushed permanently

/PROXY	Used with /ADD to indicate that the translation of the local host's Ethernet address should be published on behalf of another host
/PUBLISH	Used with /ADD to indicate that the translation to be added is to be published on behalf of another host
/TEMPORARY	Used with /ADD or /FLUSH to specify whether an entry should be added or flushed temporarily. This is the default.

# GIF (Generic/Gateway) Interface Usage

The GIF interface allows for the creation of Virtual Private Networks (VPNs) by encapsulating the traffic directed to the interface's remote address to within an additional IP header, creating a virtual network. If the traffic over this interface is subject to IPSEC, then the virtual network is private.

Each GIF interface has four IP addresses that need to be configured:

1. The local address for the interface
2. The remote (point to point peer) address for the interface
3. The gateway address for this side of the tunnel
4. The destination address for the remote side of the tunnel.

The GIF is configured with the following commands on the local system:

```
$ MULTINET SET/INTERFACE/CREATE GIFn
$! n is unit number, compile time limited
$ MULTINET SET/INTERFACE GIFn/PROTOCOL=IP/ADDRESS=A.B.C.D
/POINT_TO_POINT=E.F.G.H
$ MULTINET SET/ROUTE/ADD=(DESTINATION=A.B.C.D,GATEWAY=127.0.0.1)
$ MULTINET SET/ROUTE/ADD=(DESTINATION=E.F.G.H,GATEWAY=A.B.C.D)
$ MULTINET SET/INTERFACE GIFn/TUNNEL=(DESTINATION=I.J.K.L, GATEWAY=M.N.O.P)
```

remote system:

```
$ MULTINET SET/INTERFACE/CREATE GIFn
$! n is unit number, compile time limited
$ MULTINET SET/INTERFACE GIFn/PROTOCOL=IP/ADDRESS=E.F.G.H
```

```
/POINT_TO_POINT=A.B.C.D
```

```
$ MULTINET SET/ROUTE/ADD=(DESTINATION=E.F.G.H,GATEWAY=127.0.0.1)
```

```
$ MULTINET SET/ROUTE/ADD=(DESTINATION=A.B.C.D,GATEWAY=E.F.G.H)
```

```
$ MULTINET SET/INTERFACE GIfn/TUNNEL=(DESTINATION=M.N.O.P, GATEWAY=I.J.K.L)
```

M.N.O.P is a public IP address (interface) on the local system. I.J.K.L is a public IP address (interface) on the remote system. A.B.C.D is the private network address on the local system. E.F.G.H is the private network address on the remote system. Routing can be set up to pass traffic for other systems through the tunnel. A command procedure could be written to create the tunnel and be used on each side with some minor exchanging of parameters. IPSEC traffic could be statically configured, or managed with the RACOON IPSEC Daemon.

To get rid of the tunnel:

```
$ MULTINET SET/INTERFACE/DELETE GIfn !delete tunnel and interface
```

```
$ MULTINET SET/ROUTE/DELETE=(DESTINATION=A.B.C.D, GATEWAY=127.0.0.1)
```

The VPN encapsulates IPv4 traffic within another IPv4 packet (RFC 1853, RFC 2003).

This VPN is not compatible with Microsoft VPN which uses either PPTP (Microsoft Proprietary) or L2TP/IPSec (RFC 2661).

# 9. Configuring MultiNet Services

This chapter describes how to configure MultiNet services. The chapter provides specific information about configuring the RLOGIN, RSHELL, NTY, TFTP, and SYSLOG services. The remaining chapters in this guide provide specifics about configuring other services.

MultiNet services require basic IP connectivity on your system for the services to function properly. Read Chapter 8 to learn how to establish IP connectivity.

For a list of the servers you can configure with `SERVER-CONFIG`, see Appendix A.

## Introducing Service Configuration

When configured, MultiNet services start when a request for service is accepted by the master server (`MULTINET_SERVER`) process, which listens for incoming connections. When a connection request is received, either a separate, detached process is created to handle the connection, or the `MULTINET_SERVER` process handles the service internally.

Configuration information for the master server is in the configuration file `MULTINET:SERVICES.MASTER_SERVER` which is read by the `MULTINET_SERVER` process when it starts to determine its configuration.

When MultiNet is first installed, the normal set of services is enabled. Before starting MultiNet, you may want to verify that the server configuration meets your needs. This may require:

- Disabling servers your site does not want accessed
- Enabling servers your site wants accessed
- Adding servers specific to your site

The server configuration supplied with MultiNet is adequate to get a system up and running. The server configuration can be easily changed as needed.

Generally, services are configured with the `SERVER-CONFIG` utility, invoked with the `MULTINET CONFIGURE /SERVERS` command. This chapter describes how to use this utility and provides information about other MultiNet servers.



# Using SERVER-CONFIG to Configure Services

The MultiNet command line-based server configuration utility (SERVER-CONFIG) is an interactive utility that controls which network servers are available on the local node. In addition, you can use SERVER-CONFIG to set restrictions on a server to prevent access from unauthorized sites, to keep a log file of connections to a server, and to limit the system resources available to a server.

## Invoking SERVER\_CONFIG

To invoke SERVER-CONFIG, enter this command:

```
$ MULTINET CONFIGURE /SERVERS
```

Exit this utility with the EXIT or QUIT command.

To display the list of servers available under MultiNet, run SERVER-CONFIG, and issue the SHOW command. You can use the SHOW /FULL command to display the characteristics of a particular server. To change the default configuration, enable a server with the ENABLE command and modify server configuration parameters with the SELECT and SET commands.

In the following example, SERVER-CONFIG enables the TFTP server (which is disabled by default) displays the TFTP server's configuration, and restarts the MULTINET\_SERVER process so the changes take effect immediately.

```
$ MULTINET CONFIGURE /SERVERS
```

```
MultiNet Server Configuration Utility 5.6(nnn)
[Reading in configuration from MULTINET:SERVICES.MASTER_SERVER]
SERVER-CONFIG>ENABLE TFTP
SERVER-CONFIG>SHOW TFTP/full
Service "TFTP":
    UDP socket (AF_INET,SOCK_DGRAM), Port 69
    INIT() = Merge_Image
    Program = "MULTINET:LOADABLE_TFTP.EXE"
SERVER-CONFIG>RESTART
Configuration modified, do you want to save it first?[YES]Return
[Writing configuration to MULTINET_COMMON_ROOT:[MULTINET]
SERVICES.MASTER_SERVER]
%RUN-S-PROC_ID, identification of created process is 20600046
```

```
SERVER-CONFIG>EXIT  
[Configuration not modified, so no update needed]
```

## SERVER-CONFIG Commands

The below table lists the SERVER-CONFIG commands.

Before using a SET command, use the SELECT command to select a service.

Command	Description
ADD	Adds a service to the configuration
ATTACH	Detaches a terminal from calling process and attaches it to another process
COPY	Copies a service to create a new service
DELETE	Deletes a service from the current configuration
DISABLE	Disables a service in the current configuration
ENABLE	Enables a service in the current configuration
EXIT	Exits from the SERVER-CONFIG session
GET	Reads in a server configuration file
NETCONTROL	Contacts the NETCONTROL server
PUSH	Accesses the DCL command line while pausing SERVER-CONFIG
QUIT	Exits SERVER-CONFIG without saving changes
RESTART	Restarts the master server process
SAVE	Writes out the current server configuration file

SELECT	Selects a server for SET command
SET ACCEPT-HOSTS	Specifies hosts that can access the server
SET ACCEPT-NETS	Specifies networks that can access the server
SET BACKLOG	Specifies server connection queue limits
SET CONNECTED	Specifies a connection-request-received routine
SET DISABLED-NODES	Specifies VMScLuster nodes on which the service is disabled
SET ENABLED-NODES	Specifies VMScLuster nodes on which the service is enabled
SET FLAGS	Specifies a flag bit mask for service operation control
SET INIT	Specifies an initialize-service routine
SET KEEPALIVE-TIMERS	Specifies probes for cleaning up dormant connections
SET LISTEN	Specifies a listen-for-connections routine
SET LOG-ACCEPTS	Enables/disables successful connections logging
SET LOG-FILE	Specifies a log message destination
SET LOG-REJECTS	Enables/disables failed connections logging
SET MAX-SERVERS	Specifies the maximum number of processes
SET PARAMETERS	Specifies service-dependent parameters; affects the current configuration and becomes active the next time MULTINET_SERVER starts
SET PQL-ASTLM	Specifies the OpenVMS AST (asynchronous system trap) limit (the number of pending ASTs available to a process)

SET PQL-BIOLM	Specifies the OpenVMS buffered I/O limit (the number of outstanding buffered I/O requests available to a process)
SET PQL-BYTLM	Specifies the OpenVMS buffered I/O byte count limit (the number of bytes allowed in any single buffered I/O request)
SET PQL-CPULM	Specifies the OpenVMS CPU time limit of the created process
SET PQL-DIOLM	Specifies the OpenVMS direct I/O limit (the number of outstanding direct I/O requests available to a process)
SET PQL-ENQLM	Specifies the OpenVMS enqueue limit of the created process
SET PQL-FILLM	Specifies the OpenVMS open file limit (the number of open files available to a process)
SET PQL-JTQUOTA	Specifies the OpenVMS job-wide logical name table byte quota (the quota allocated to the job-wide logical name table on its creation)
SET PQL-PGFLQUOTA	Specifies the OpenVMS paging file quota of the created process
SET PQL-PRCLM	Specifies the OpenVMS sub-process limit (the number of sub-processes available to a process)
SET PQL-TQELM	Specifies the OpenVMS timer queue entry limit (the number of timer queue entries available to a process)
SET PRIORITY	Specifies the OpenVMS priority for created processes
SET PROGRAM	Specifies an OpenVMS file name for run or merged images
SET RECEIVE-BUFFER-SIZE	Specifies the size of receive socket buffer
SET REJECT-BY-DEFAULT	Specifies what happens if no match is found on SET ACCEPT-HOSTS and SET REJECT-HOSTS, and on SET ACCEPT-NETS and SET REJECT-NETS

SET REJECT-HOSTS	Specifies hosts not allowed service access
SET REJECT-MESSAGE	Specifies a rejected connection message
SET REJECT-NETS	Specifies networks not allowed service access
SET SEND-BUFFER-SIZE	Specifies the size of the send socket buffer
SET SERVICE	Specifies a perform-service routine
SET SERVICE-NAME	Changes a service name. The underscore character can be used in the service name.
SET SOCKET-FAMILY	Specifies service family address
SET SOCKET-OPTIONS	Specifies <code>setsockopt ()</code> options
SET SOCKET-PORT	Specifies the port for connection listening
SET SOCKET-TYPE	Specifies the socket type
SET USERNAME	Specifies the name of the user under which the service is started; applies only to UCX services. A UCX service has the <code>UCX_SERVER</code> flag set
SET WORKING-SET-EXTENT	Specifies the OpenVMS working set extent of the created process
SET WORKING-SET-QUOTA	Specifies the OpenVMS working set quota of the created process
SHOW	Shows the current server configuration
SHUTDOWN	Stops the master server process
SPAWN	Executes a DCL command, or mimics <code>PUSH</code>
STATUS	Shows the <code>SERVER-CONFIG</code> status

USE	Reads in a server configuration file
VERSION	Shows the SERVER-CONFIG version
WRITE	Writes out the current server configuration file

When you run SERVER-CONFIG commands, a number of prompts are displayed. These prompts are explained in Appendix A. There are a number of per-service prompts when you modify server parameters. Some of these parameters control the operation of the server, including process priority, working set limit, restrictions, and auditing. Other parameters define operations of the server, such as the protocol and port number.

## Adding Your Own Services

The MULTINET\_SERVER process can listen for user-written services (including IPX/SPX) and, when a connection request arrives, create an OpenVMS detached process running the user-written program. This process is created with full privileges. SYS\$INPUT, SYS\$OUTPUT, and SYS\$ERROR are set to the network. See the *MultiNet Programmer's Reference* for descriptions of library routines for writing your own services that interface to MultiNet.

The following example shows how to add a user-written service called NNTP to the MultiNet configuration. The program NNTP\_SERVER.EXE is invoked when an NNTP connection arrives.

```
SERVER-CONFIG>ADD NNTP
[Adding new configuration entry for service "NNTP"]
Protocol: [TCP] TCP
TCP Port number: 119
Program to run: USER$DISK: [NNTP]NNTP_SERVER.EXE
[Added service NNTP to configuration]
[Selected service is now NNTP]
SERVER-CONFIG>
```

**Note:** If your service uses the `getservbyname ( )` or `getservbyport ( )` socket library functions, you must also add your service to the `HOSTS.LOCAL` file and recompile your host tables.

# Disabling, Enabling, and Deleting Services

You can tailor the MULTINET\_SERVER to meet your specific needs by enabling or disabling services using the ENABLE, DISABLE, or DELETE commands.

For example, to enable BOOTP service with the SERVER-CONFIG ENABLE BOOTP command and restart the server to make the change immediately available, issue these commands:

```
$ MULTINET CONFIGURE /SERVER
MultiNet Server Configuration Utility 5.6(nnn)
[Reading in configuration from MULTINET:SERVICES.MASTER_SERVER]
SERVER-CONFIG>ENABLE BOOTP
SERVER-CONFIG>RESTART
Configuration modified, do you want to save it first ? [YES] RETURN
[Writing configuration to MULTINET_COMMON_ROOT:[MULTINET]
SERVICES.MASTER_SERVER]
%RUN-S-PROC_ID, identification of created process is 20600046
SERVER-CONFIG>EXIT
[Configuration not modified, so no update needed]
$
```

The following example shows how to disable the SMTP service:

```
SERVER-CONFIG>DISABLE SMTP
```

## Disabling or Enabling Services on a Per-Cluster-Node Basis

You can enable or disable services on a per-node basis in a VMScLuster. Using the SET ENABLED-NODES or SET DISABLED-NODES commands, you can specify a list of VMScLuster nodes on which the service does or does not run. You must also enable the service using the ENABLE command. For example, to enable the SMTP service to run only on the VMScLuster node VMSA:

```
SERVER-CONFIG>SELECT SMTP
[The Selected SERVER entry is now SMTP]
SERVER-CONFIG>SET ENABLED-NODES
You can now add new VAXcluster nodes for SMTP.
An empty line terminates.
Add VAXcluster node: VMSA
```

```
Add VAXcluster node: RETURN  
SERVER-CONFIG>
```

# Restricting Access to Servers

MultiNet allows a system manager to restrict access to services on a per-service, per-network, or per-host basis.

**Note:** Restriction lists are supported only for services that listen for connections through `MULTINET_SERVER`. For example, the NFS Server, which reads datagrams directly, ignores any restrictions configured through `SERVER-CONFIG`.

Five service parameters (`ACCEPT-HOSTS`, `ACCEPT-NETS`, `REJECT-HOSTS`, `REJECT-NETS`, and `REJECT-BY-DEFAULT`) control whether the `MULTINET_SERVER` process allows or rejects a connection request based on the requesting host address, network, or subnetwork.

- If the connection comes from a host listed in the `ACCEPT-HOSTS` or `REJECT-HOSTS` lists, the connection is accepted or rejected, respectively.
- If the host is not found in one of these lists, the `ACCEPT-NETS` and `REJECT-NETS` lists are examined.
- If a match is found, the connection is accepted or rejected.
- If the host does not match any of these four lists, the action taken is governed by the `REJECT-BY-DEFAULT` parameter, which is normally set to `FALSE`, indicating that all connections are accepted.

You can also use the `ACCEPT-NETS` and `REJECT-NETS` parameters to specify a subnetwork number. You specify a subnetwork by supplying the subnetwork number followed by a space and the subnetwork mask for that subnetwork. For example:

```
REJECT-NETS 128.1.1.0 255.255.255.0
```

rejects only the hosts on the 128.1.1.n network. All other hosts on 128.1.n.n have access.



**Note:** If you answer YES after the Internet address at the prompt, only connections from a port number below 1024 are accepted.

The action taken to reject a connection depends on the protocol involved; for example, a UDP datagram is rejected by ignoring it, but a TCP connection is rejected by immediately closing the connection.

The REJECT-MESSAGE parameter specifies a text string sent to the client before the connection is closed. The following example shows how to restrict access to the TELNET server to hosts that are on a local network at address 128.1.0.0, with the one exception of the host at the address 192.0.0.2.

```
$ MULTINET CONFIGURE/SERVERS
MultiNet Server Configuration Utility 5.6(nnn)
[Reading in configuration from MULTINET:SERVICES.MASTER_SERVER]
SERVER-CONFIG>SELECT TELNET
[The Selected SERVER entry is now TELNET]
SERVER-CONFIG>SET ACCEPT-NETS
You can now add new addresses for TELNET.  An empty line terminates.
Add Address: 128.1.0.0
Add Address: RETURN
SERVER-CONFIG>SET ACCEPT-HOSTS
You can now add new addresses for TELNET.  An empty line terminates.
Add Address: 192.0.0.2
Add Address: RETURN
SERVER-CONFIG>SET REJECT-BY-DEFAULT TRUE
SERVER-CONFIG>SET REJECT-MESSAGE Illegal source of TELNET connection
SERVER-CONFIG>SHOW/FULL
Service "TELNET":
    TCP socket (AF_INET,SOCK_STREAM), Port 23
    Socket Options = SO_KEEPALIVE
    INIT() = TCP_Init
    LISTEN() = TCP_Listen
    CONNECTED() = TCP_Connected
    SERVICE() = Internal_Telnet
    Accept Hosts = IP-192.0.0.2
    Accept Nets = IP-128.1.0.0
    Reject by default all other hosts and nets
    Reject Message = "Illegal source of TELNET connection"
SERVER-CONFIG>RESTART
Configuration modified, do you want to save it first ? [YES] YES
[Writing configuration to
MULTINET_COMMON_ROOT:[MULTINET]SERVICES.MASTER_SERVER]
%RUN-S-PROC_ID, identification of created process is 20600054
SERVER-CONFIG>
```

# Auditing Access to Servers

MultiNet allows the security-conscious system manager to audit access to a service, file, or the OpenVMS process. Three service parameters govern the auditing that occurs when a connection is accepted or rejected:

LOG-ACCEPTS	Enables or disables logging for accepted connection requests.
LOG-FILE	Specifies the OpenVMS file name to which log messages are written. The auditing data collected by the MultiNet Server is collected in this file and flushed (written to disk) approximately every five to ten minutes. To maintain different versions of this file, you can copy an empty file over the old one on a daily basis (or at a frequency that meets your needs).
LOG-REJECTS	Enables or disables logging for rejected connection requests. A request can be rejected if the REJECT-HOSTS, REJECT-NETS, or REJECT-BY-DEFAULT security restrictions are enabled and succeed, or if the ACCEPT-HOSTS and ACCEPT-NETS restrictions are enabled and fail. For more information, refer to the <i>Restricting Access to Servers</i> section.

**Note:** The only services that support auditing are those that listen for connections through the MULTINET\_SERVER. For example, the optional NFS server, which reads datagrams directly, ignores the auditing parameters.

You should not turn on logging for a UDP service. Because there is no "formal" connection for UDP, every packet sent to the UDP service would be logged.

The following example shows how to enable a log file on the TELNET service.

```
$ MULTINET CONFIGURE /SERVERS
MultiNet Server Configuration Utility 5.6 (nnn)
[Reading in configuration from MULTINET:SERVICES.MASTER_SERVER]
SERVER-CONFIG> SELECT TELNET
[The Selected SERVER entry is now TELNET]
SERVER-CONFIG> SET LOG-ACCEPTS TRUE
SERVER-CONFIG> SET LOG-REJECTS TRUE
```

```

SERVER-CONFIG>SET LOG-FILE MULTINET:SERVER.LOG
SERVER-CONFIG>SHOW/FULL
Service "TELNET":
TCP socket (AF_INET,SOCK_STREAM), Port 23
Socket Options = SO_KEEPALIVE
INIT() = TCP_Init
LISTEN() = TCP_Listen
CONNECTED() = TCP_Connected
SERVICE() = Internal_Telnet
Log File for Accepts & Rejects = MULTINET:SERVER.LOG
SERVER-CONFIG>RESTART
Configuration modified, do you want to save it first ? [YES] YES
[Writing configuration to MULTINET_COMMON_ROOT:[MULTINET]
SERVICES.MASTER_SERVER]
%RUN-S-PROC_ID, identification of created process is 20600054
SERVER-CONFIG>

```

The next example shows the auditing records written to the log file.

```

15-JUN-2015 17:27:00 RPCPORTMAP (accepted) from [127.0.0.1,108] (localhost)
15-JUN-2015 17:50:25 FINGER (accepted) from [10.1.228.65,1071] (ABC.COM)
15-JUN-2015 21:10:40 RLOGIN (accepted) from [10.1.228.65,1022] (ABC.COM)
16-JUN-2015 11:49:46 FINGER (accepted) from [10.1.228.65,1214] (ABC.COM)
16-JUN-2015 11:51:05 RLOGIN (accepted) from [10.1.228.68,1022] (Bubba)
16-JUN-2015 20:09:48 FTP (accepted) from [10.1.228.68,1039] (Bubba)

```

## Writing an Auditing Dispatcher

MultiNet allows a system manager to further customize the auditing facilities.

You can provide a user-written shareable image that is merged into the MULTINET\_SERVER when the server is restarted. The user-written shareable image is called whenever a connection arrives with information about the connection and can perform the desired auditing.

If the MultiNet Programmer's Kit is installed, the directory MULTINET\_ROOT:[MULTINET.EXAMPLES] contains the file USER\_AUDITOR.C. This is a C-language template of a routine called when a connection arrives.

Compile, link, and place it in the MULTINET: directory according to the instructions at the beginning of the file.

# Detecting Intruders

MultiNet provides the IP address and an optional IP port number of a suspected intruder to OpenVMS accounting and intrusion detection. The IP address is recorded in hexadecimal format to make room for the optional IP port address. By default, the IP port address is included. To disable this feature, set the LGI\_BRK\_TERM system parameter to zero.

This example shows accounting and intrusion reports generated with LGI\_BRK\_TERM set to one (the default).

```

$ ACCOUNTING/NODE=TELNET
Remote node name:  TELNET           Privilege <31-00>: 00148000
Remote ID:         A12C800C:0F94    Privilege <63-32>: FFFFFFFC0
                   ^^^^^^^^^^  ^^^^^
IP address IP port
$ SHOW INTRUSION
Intrusion  Type          Count      Expiration      Source
NETWORK    SUSPECT          2          17:35:09.28     TELNET::A12C800C:0F94
                   ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
                                           IP address IP port

```

In these examples, A12C800C is the IP address 10.1.34.22 and 0F94 is IP port 3988, both expressed in hexadecimal. You can use the following DCL code to convert hexadecimal IP addresses and port numbers (of the form hex-address:hex-port-number) into dotted decimal format:

```

$ IP_ADDRESS = F$ELEMENT(0,":",P1)
$ PORT_NUMBER = F$ELEMENT(1,":",P1)
$ WRITE SYS$OUTPUT F$FAO("IP address/port: !UL.!UL.!UL.!UL!/!UL", -
F$INTEGER("%X'"F$EXTRACT(0,2,IP_ADDRESS)'"'), -
F$INTEGER("%X'"F$EXTRACT(2,2,IP_ADDRESS)'"'), -
F$INTEGER("%X'"F$EXTRACT(4,2,IP_ADDRESS)'"'), -
F$INTEGER("%X'"F$EXTRACT(6,2,IP_ADDRESS)'"'), -
F$INTEGER("%X'"PORT_NUMBER'"'))
$ EXIT

```

Intrusion detection uses the physical (Ethernet) address, SPX port number, and possibly the target user name. The target user name appears only when the break-in attempt targets a valid account and LGI\_BRK\_TERM is set to one (the default), as in the following example intrusion report.

```

$ SHOW INTRUSION
Intrusion  Type          Count      Expiration      Source
TERM_USER  SUSPECT          1          16:23:38.67     [A12C8000:AA.00.04.00.08.60#ECB]:
                   ^^^^^^^^^^  ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^  ^^^  ^^
                                           Network Ethernet address SPX User
                                           number                    port
Name

```

In this example,

- A12C8000 is the network number, expressed in hexadecimal.

- AA.00.04.00.08.60 is the Ethernet address, expressed in dotted-hexadecimal format.
- ECB is the SPX port.
- No user name is specified; the user name normally follows the ending colon.

The OpenVMS SHOW INTRUSION Utility truncates the Source display field to 33 characters. The TERM\_USER record is actually associated with a particular user name; you can delete it (with DELETE/INTRUSION) only by specifying the full source specification, including the invisible, truncated data.

## Detecting Intruders on an FTP Server

The following example of an FTP\_SERVER.COM file shows the use of the FTP server qualifiers. The first section ensures that global and local symbols appear as expected. Information is then taken from the logical names that store data about the person who has accessed this FTP server:

MULTINET_FTP_ADDRESS	Provides an ASCII representation of the user's IP address.
MULTINET_FTP_HOSTNAME	Provides an ASCII representation of the user's host name.
MULTINET_ANONYMOUS_PASSWORD	Contains an ASCII representation of the user's anonymous FTP password.
MULTINET_FTP_MAXIMUM_IDLE_TIME	Controls the duration (in seconds) the FTP server allows a connection to be idle before it is closed. The default is 300 seconds. This logical can be specified in any logical name table accessible to the user running the FTP server.

```
$ Set := "Set"
$ Set Symbol/Scope=(NoGlobal,NoLocal)
$ If F$TrnLNM("MULTINET_FTP_ADDRESS") .Eqs. "" Then -
MultiNet FTP/Server/Get_Remote_Info
$ FTP_Address = F$TrnLNM("MULTINET_FTP_ADDRESS")
$ FTP_Hostname = F$TrnLNM("MULTINET_FTP_HOSTNAME")
$ FTP_Password = F$TrnLNM("MULTINET_ANONYMOUS_PASSWORD")
$ Ident = FTP_Hostname
$ If FTP_Hostname .Eqs. "" Then Ident = FTP_Address
```

In the next section, if FTP\_Hostname is null, the IP address could not be found in the host table or by DNS lookup. The Ident symbol is set to flag this event:

```

$ Message = ""
$ If FTP_Hostname .Eqs. "" Then -
    Message = Message + ", ""Unknown hostname: ''Ident'; DNS Problem!""
$ If F$Edit(FTP_Password, "UPCASE") .Eqs. "GUEST" Then -
    Message = Message + ", ""''Ident'; say who really you are.""
$ Message = Message + ", ""@WELCOME.TXT""
$ If Message .Nes. "" Then -
    Message = "/Message=( "+F$Extract(1,256,Message)+") "

```

In the following section, the banner message is altered to fit the circumstances under which the user logs in.

```

$ If F$Extract(0,6,FTP_Address) .Eqs. "192.0." Then Goto Intruder
$!
$ DirOptions = "/Directory=Users:[Anonymous]"
$ AccOptions = "/Access=NoWrite"
$ MultiNet FTP/Server 'AccOptions' 'DirOptions' 'Message'
$!
$ Logout/Brief

```

The following section tests the IP address for a known intruder. If the intruder is discovered, control is passed to the `Intruder` label. Next, the FTP server is called to handle the anonymous login, and when done, logs out.

1. An offensive stance is taken when an intruder is discovered by running a FINGER of the intruder's system.
2. `INFLAME.TXT` is invoked to display a personalized message.
3. The FINGER output is displayed on the intruder's terminal.
4. Mail is sent to the system manager to indicate that an intrusion event occurred.
5. The procedure exits.

```

$ Intruder:
$ Set NoOn
$ create finger.temp
$ define/user sys$output finger.temp
$ MultiNet Finger @'FTP_Address
$!
$ MultiNet FTP/Server/Reject/Message="@Inflame.txt",-
"Thanks for listening ''FTP_Password'@''Ident';
    now smile:", "", "@finger.temp")
$ Mail/Subject="intruder FTP access from
    ''FTP_Password'@''Ident'" finger.temp system
$ del finger.temp;*
$ Logout/Brief

```

# Detecting Intruders with NETCONTROL Accounting

If OpenVMS Accounting is enabled on your system, the following process header fields are set for network server processes created by the MultiNet master server:

- `CTL$T_NODENAME` is the name of the service being run; for example, FTP, TELNET, or SMTP.
- `CTL$T_REMOTEID` is the IP address of the remote client system in dotted-decimal format.

This change can make attempted break-ins to your system easier to track. It also provides a simple mechanism for tracking remote access to your system.

For example, to determine which users are using TELNET to gain access to your system, you can issue the command:

```
$ ACCOUNTING /NODE=TELNET
```

**Note:** IP address and port information is displayed in hexadecimal in accounting reports.

The `MULTINET NETCONTROL` command features `ACCOUNTING` and `DEBUG` parameters for the `NETCONTROL` (master server) service.

The format of this command is:

```
$ MULTINET NETCONTROL NETCONTROL ACCOUNTING n
```

By default (as described above), additional information is provided in the accounting record by the MultiNet server. You can disable this feature by setting `n` to 0. When set to 1, the remote name and service name are added to the `ACCOUNTING` record.

OpenVMS adds accounting records to a central database each time a special system event occurs. These events include processes exiting, images (programs) ending, failed logins, and so on. OpenVMS stores a record of the event with information from the process.

One item that OpenVMS includes in the accounting record is the remote node ID, which it gets from the internals of the process. For most processes, the remote node ID is empty.

When you use `SET HOST` to create a log for another node, DECnet fills in the remote node ID field when it creates the process. This information then appears in any accounting record generated for that process.

If you suspect an intruder has attempted a security breach of your system, you can examine the accounting records to see who has logged in and identify where a login attempt originated.

The master server fills in both the remote node ID and the node name field. The remote node ID field is set to the ASCII representation of the dotted-decimal IP address of the node that requested the service. The NODE NAME field is set to the service name, such as FTP, TELNET, and so on.

The following example shows a LOGIN failure accounting record:

```
LOGIN FAILURE
-----
Username:          JDOE          UIC:                [SYSTEM]
Account:          <net>         Finish time:        23-MAR-2020 21:27:34.47
Process ID:       21200124      Start time:         23-MAR-2020 21:27:33.81
Owner ID:         0             Elapsed time:       0 00:00:00.66
Terminal name:    Processor time: 0 00:00:00.26
Remote node addr: Priority:          4
Remote node name: FTP          Privilege <31-00>: FFFFFFFF
Remote ID:        10.1.1.94    Privilege <63-32>: FFFFFFFF
Remote full name:
Queue entry:      Final status code: 00D380FC
Queue name:
Job name:
Final status text:%LOGIN-F-INVPWD, invalid password
Page faults:     166          Direct IO:          7
Page fault reads: 5          Buffered IO:        12
Peak working set: 246          Volumes mounted:   0
Peak page file:  3280         Images executed:    1
```

## Using UCX-Compatible Services under MultiNet

Services written to be started by the auxiliary server (INETD) in HP TCP/IP Services can be configured for use with MultiNet by setting the service parameter SET FLAGS UCX\_SERVER.

Driver enhancements support TeamLinks, POSIX sockets, DCE for OpenVMS, and TCP/IP Services V2.0 keepalive compatibility.

The logical names UCX\$INET\_HOSTADDR and TCPIP\$INET\_HOSTADDR contain the text value of the primary interface. MultiNet defines UCX\$INET\_HOSTADDR and TCPIP\$INET\_HOSTADDR automatically, much the same as other TCP/IP Services logical names. It is defined in START\_MULTINET.COM.



Performing a close (das sgn) operation on any TCP/IP Services (BG) device used in a select list cancels the select operation.

The following logicals are defined for improved UCX compatibility:

```
TCPIP$BIND_DOMAIN
TCPIP$BIND_SERVER00
TCPIP$BIND_SERVER001 (if more than one BIND server is defined)
TCPIP$BIND_SERVER002 (if more than one BIND server is defined)
TCPIP$DEVICE = BG:
TCPIP$INET_DEVICE = BG:
TCPIP$INET_HOST
TCPIP$INET_HOSTADDR
TCPIP$IPC_SHR = MULTINET:UCX$IPC_SHR
```

## Associating Command Procedures with Services

You can specify DCL command procedures (.COM files) as the programs associated with MultiNet services. When a service is initiated, MultiNet calls LOGINOUT to invoke the user's LOGIN.COM file and the specified DCL command procedure. When called, the DCL and CLI are mapped for use by the process. This feature gives the system manager a "hook" into a service with an easy-to-create command procedure. Note, however, that the command procedure cannot use SYS\$INPUT. Therefore, do not use the READ SYS\$INPUT or INQUIRE commands in the command procedure or a user's LOGIN.COM file.

Make sure that all command procedures associated with services include a command that assigns SYS\$INPUT to SYS\$OUTPUT as follows:

```
$ DEFINE /USER SYS$INPUT SYS$OUTUT
```

This command must appear immediately before any command that runs an image, but is only in effect while the image is running. To ensure the assignment lasts for the duration of the entire command procedure, use the above command without the /USER qualifier:

```
$ DEFINE SYS$INPUT SYS$OUTPUT
```

# Setting Keepalive Timers

Keepalives are useful when other systems that connect to services provided by your system are subject to frequent crashing, resets, or power-offs (as with personal computers). TCP/IP connections must normally pass through a three-way handshake sequence to be closed and removed from the connection table. If a connection is open but idle, and the remote system is shut down, reset, or crashes, the connection is not closed down until your system attempts to communicate with the remote system. If an application or service does not attempt to communicate, a keepalive probe can clean up these dormant connections.

The format for the `SET KEEPALIVE-TIMERS` function is:

```
SERVER-CONFIG>SELECT service  
SERVER-CONFIG>SET KEEPALIVE-TIMERS idle-time probe-interval probe-count
```

<i>idle-time</i>	is the amount of time, in seconds, that a connection should be idle before the first keepalive probe is sent.
<i>probe-interval</i>	is the number of seconds between keepalive probes (75 seconds minimum).
<i>probe-count</i>	is the number of probes sent, with no reply from the other side of the connection, before the connection is destroyed.

Setting any of these parameters to 0 retains its default setting.

If you set the `SO_KEEPALIVE` socket option for a service, but you do not explicitly set `KEEPALIVE-TIMERS`, the default values are:

- `idle-time` is 2 hours.
- `probe-interval` is 75 seconds.
- `probe-count` is 8.

If you do not set the `SO_KEEPALIVE` socket option for a service, no keepalive probes are sent for connections to that service.

# Configuring TFTP (Trivial File Transfer Protocol)

The MultiNet TFTP service uses standard Internet TFTP to perform file transfers. Like FTP, TFTP can also be used to transfer files between a host running OpenVMS and a remote host. Unlike FTP, TFTP cannot perform operations other than transferring files between a local system and a remote one; that is, TFTP cannot list directories, delete files, and so on.

TFTP does not perform any authentication when transferring files, so a user name and password on the remote host are not required. In general, only files with world-read (W:R) access in certain directories on the remote host are available for reading, and only certain directories are available for writing.

Use `SERVER-CONFIG` to enable TFTP as follows:

```
$ MULTINET CONFIGURE /SERVER
MultiNet Server Configuration Utility 5.6(nnn)
[Reading in configuration from MULTINET:SERVICES.MASTER_SERVER]
SERVER-CONFIG>ENABLE TFTP
SERVER-CONFIG>EXIT
[Writing configuration to MULTINET_COMMON_ROOT:[MULTINET] |
SERVICES.MASTER_SERVER]
$
```

**Note:** The permissions of the accessed directories are not checked before the access is attempted. Because the TFTP protocol does not specify any user login or validation, the TFTP server permits only WORLD-readable files to be accessed.

The TFTP server normally requires full file system pathnames, as it operates without a default directory. If you are using TFTP to download network servers which may not be able to provide full pathnames, you can set a TFTP default directory using the `NET-CONFIG SET TFTP-DIRECTORY` command.

**Note:** The TFTP mail option, as defined in RFC-783, is obsolete and not supported under the MultiNet TFTP server.

**Note:** The TFTP server supports the BLKSIZE option (RFC 2348) with a minimum of 512 and maximum of 8192. For image (octet) transfers the size must be a multiple of 512.

## TFTP File Name Translations

The `MULTINET:TFTP.FILENAME-TRANSLATIONS` file can be used to translate between TFTP client file names and OpenVMS file names, and restrict TFTP access to certain files or directories. The following is an example of a `MULTINET:TFTP.FILENAME-TRANSLATIONS` file:

```
#
# TFTP filename fixups for broken TFTP
# loaders & file access restrictions
#
RESTRICT-ACCESS
#
# This is a translation for a single file
#
/Foo/bar.dat                SYS$MANAGER:BAR.DAT
#
# These two translations map an entire OpenVMS directory
#
/usr/lib/X11/ncd/configs/*   NCD_CONFIGS:[000000]
/ncd_fonts/dw75dpi/*        NCD_FONTS:[DW75DPI]
#
# This translation is a file access restriction
#
SYS$SYSROOT:[SYSFONT.DECW.*  SYS$SYSROOT:[SYSFONT.DECW.
```

Use either # or ! to start a comment.

If the keyword `RESTRICT-ACCESS` is on a line by itself in the file, only files that match one of the translation specifications are accessible. This restricts TFTP access to specific directory hierarchies. If this string is not specified in the `TFTP.FILENAME-TRANSLATIONS` file, all `WORLD`-readable files are accessible to TFTP clients.

The first translation causes a client reference to `/Foo/bar.dat` to access the OpenVMS file `SYS$MANAGER:BAR.DAT` (note that comparisons are not case-sensitive). This is an example of a translation for a single file.

The next two translations are examples of mapping an entire OpenVMS directory. If the client reference matches everything up to the asterisk (\*), the rest of the client reference is appended to the translation string. For example, `/usr/lib/X11/ncd/configs/foo.dat` becomes `NCD_CONFIGS:[000000]FOO.DAT`.

The last translation is an example of a file access restriction. The result of the translation is the same as the client-specified file. The TFTP server disallows subdirectory specifications that include `.-` to ensure you cannot bypass access restrictions by going back up the VMS directory hierarchy.

If the file `MULTINET:TFTP.FILENAME-TRANSLATIONS` is edited, `NETCONTROL` must be used to `RELOAD` it:

```
$ MULTINET NETCONTROL TFTP RELOAD
```

## Configuring "R" Services

This section describes configuration of the "R" services, `RLOGIN` and `RSHELL`.

- `RLOGIN` provides a means of logging in to another system.
- `RSHELL` executes commands remotely on another system.

These services are enabled when MultiNet is installed.

The authentication scheme used by `RLOGIN` and `RSHELL` is based on trusted users and trusted hosts specified in files on the destination system. The `MULTINET:HOSTS.EQUIV` file (`/etc/hosts.equiv` on UNIX systems) grants access on a system-wide basis. The file `SYS$LOGIN:.RHOSTS` (`~/rhosts` on UNIX systems) can be used by individual users on a system to grant remote users access to their accounts.

**Note:** Do not use IP addresses in the `HOSTS.EQUIV` or `.RHOSTS` file.

Access control requirements differ between `RLOGIN` and other R services. `RLOGIN` requires both `NETWORK` and `LOCAL` access, while `RSHELL`, `REXEC`, `RMT`, and `RCP` only require `NETWORK` access.

If you remove a user's `NETWORK` access, the user can still log in until their `RLOGIN` cache entry expires or is flushed. However, if you remove that user's `LOCAL` access, the user is denied access immediately, even if they have a current cache entry.

The format of an entry in the `HOSTS.EQUIV` or `.RHOSTS` file is:

```
hostname      [username]
```

If an entry containing only the host name is in the file `MULTINET:HOSTS.EQUIV` (or `/etc/hosts.equiv`) on the target system, all users on `hostname` with the same user name as on the target system can access the target without specifying a user name or password.

Both `MULTINET:HOSTS.EQUIV` and `SYS$LOGIN:.RHOSTS` accept "wildcards" in host names. For example, to specify all hosts at `EXAMPLE.COM`, include the following line:

```
*.EXAMPLE.COM
```

If an entry in `SYS$LOGIN:.RHOSTS` or a user's `~/rhosts` file contains the following format, the specified username on the `hostname` system can access the user's account on the target system without specifying a password (or a user name if the user names are identical on the two systems):

```
hostname      username
```

The next example shows a `HOSTS.EQUIV` file on the host `SALES.EXAMPLE.COM` that gives users on `SALES.EXAMPLE.COM` `RLOGIN` and `RSHELL` access to their own accounts on the system (this is allowed by the first two entries).

In this example, `EXAMPLE.COM` and `BUBBA.EXAMPLE.COM` are identified (in the last two entries) as trusted hosts, allowing any user on either of these systems to have `RLOGIN` and `RSHELL` access to the account of the same name on `SALES.EXAMPLE.COM` without specifying a user name or password.

```
localhost
sales.example.com
example.com
bubba.example.com
```

This example shows a `.RHOSTS` file that belongs to a user on `SALES.EXAMPLE.COM`:

```
example.com      system
unix.example.com root
```

The first entry grants access to the user's account on the host `SALES.EXAMPLE.COM` to user `SYSTEM` on host `EXAMPLE.COM`. The second entry grants access to the user's account to user `root` on host `UNIX.EXAMPLE.COM`. Hence, either of these two remote users can use `RLOGIN` or `RSHELL` to access the user's account on `SALES.EXAMPLE.COM` without specifying a password.

**Note!** When specifying a user in any of the authentication files (particularly on the UNIX operating system), make sure to specify the user name in the correct case. `ROOT` and `root` are treated as different user names under case-sensitive systems such as UNIX.

The host initiating the `RLOGIN` or `RSHELL` request must be listed in the destination host's hostname database or must be resolvable within the Domain Name System (DNS), if domain name service is enabled. If the destination host cannot determine the initiating host's name from the IP address in the connection request, it rejects the request.

The RLOGIN server parameters INCLUDE-AUTHENTICATION-INDICATION, INCLUDE-PORT-NUMBER and INCLUDE-SEND-LOCATION cause an authentication indication, a port number, or a send location (or all three) to be placed in the TT\_ACCPORTNAM field of the NTY device control block. These parameters are disabled by default. To enable them, use SERVER-CONFIG (MULTINET CONFIGURE /SERVER).

## Disabling the Standard Error RSHHELL Connection

The RSHHELL /NOSTDERR option disables creation of the connection from the remote RSHHELL server back to the client for "standard error" output. This allows you to use the RSHHELL command through firewalls that do not allow TCP connections that originate from outside the local network. When you use this option, the remote RSHHELL server sends messages to the "standard output" connection, so error messages are still displayed.

## RLOGIN and RSHHELL Authentication Cache

The MultiNet RLOGIN and RSHHELL servers cache the contents of the .RHOSTS and HOSTS.EQUIV files and authentication information from the SYSUAF file in memory for ten minutes to improve performance. This means that changes made to these files may not be noticed by the network immediately. Use the following command to flush the cache before the timeout period:

```
$ MULTINET NETCONTROL RLOGIN FLUSH
```

You can use SERVER-CONFIG to change the timeout on these caches by setting the RHOSTS-TIMEOUT or UAF-TIMEOUT parameters for RSHHELL, RLOGIN, or REXEC. The default value for these parameters is 600 seconds. Setting a value of 0 (zero) disables the cache. Because the "R" services share a common authentication cache, you need only set these parameters for one service. If you set different values for different servers, only one of the values is used, so set these parameters on only one server. The following example shows how to set the timeout parameter.

```
$ MULTINET CONFIGURE/SERVER
MultiNet Server Configuration Utility 5.6(nnn)
[Reading in configuration from MULTINET:SERVICES.MASTER_SERVER]
SERVER-CONFIG>SELECT RLOGIN
[The Selected SERVER entry is now RLOGIN]
SERVER-CONFIG>SET PARAMETERS
Delete parameter "rhosts-timeout 0" ? [NO] YES
[Parameter "rhosts-timeout 0" deleted from RLOGIN]
You can now add new parameters for RLOGIN. An empty line terminates.
Add Parameter: RHOSTS-TIMEOUT 3600
```

```
Add Parameter: UAF-TIMEOUT 60
Add Parameter:
[Service specific parameters for RLOGIN changed]
SERVER-CONFIG>
```

The UAF access type specification for RSHELL access is NETWORK. MultiNet checks all access times.

The MULTINET NETCONTROL RLOGIN SHOW command shows the time the information read in from specific .RHOSTS files will expire. This is the time remaining until the .RHOSTS file is read in again.

**Note:** A non-existent .RHOSTS file is treated the same as an empty .RHOSTS file.

## Controlling RSHELL and REXEC Process Deletion

If a client closes a connection before the remote process finishes, the RSHELL and REXEC servers may delete the process. This behavior affects PC-based X servers that use REXEC to launch X applications.

This default action can be changed by using SERVER-CONFIG to SET PARAMETER as in the following example:

```
$ MULTINET CONFIGURE /SERVER
. . . startup messages . . .
SERVER-CONFIG>select rshell
[The Selected SERVER entry is now RSHELL]
SERVER-CONFIG>set parameter
You can now add new parameters for RSHELL. An empty line terminates.
Add Parameter: ?
parameter, one of the following:
DEBUG          DISALLOW-RHOSTS          DISALLOW-X-DISPLAY
PREVENT-PROCESS-DELETION  RHOSTS-TIMEOUT          UAF-TIMEOUT
or confirm with carriage return
Add Parameter: PREVENT-PROCESS-DELETION
Add Parameter:
[Service specific parameters for RSHELL changed]
SERVER-CONFIG>
```



## Controlling Automatic WSA Device Creation

By default, MultiNet R services create WSA devices and set displays to simplify setup for X client users, allowing users to run X clients without explicitly issuing the `SET DISPLAY` command. To disable this feature, set `DISALLOW-X-DISPLAY` with the `SET PARAMETER` command (in `SERVER-CONFIG`).

## Inhibiting Output in Command Procedures for "R" Services

Problems arise when remote users log into systems using a login command procedure (`SYS$LOGIN:SYLOGIN.COM` or `SYS$MANAGER:SYLOGIN.COM`) that requires screen output. To inhibit this behavior, make sure the following lines are included at the top of all login command procedures:

```
$ VERIFY = 'F$VERIFY(0)           ! Turn off verify without echoing
$ IF F$MODE() .EQS. "OTHER" THEN EXIT ! If a DETACHED process (RSHELL)
. . .
$ IF VERIFY THEN SET VERIFY       ! If a batch job, may want to turn
                                   ! verify back on.
```

## Permitting "R" Service Access to Captive or Restricted Accounts

In general, R services should not be permitted access to captive or restricted OpenVMS accounts. However, if your users depend on such access, define the following logical to allow access to these types of accounts:

```
$ DEFINE/SYSTEM/EXEC MULTINET RSHELL ALLOW CAPTIVE "TRUE"
```

## Configuring the TELNET Server for Kerberos V5

Enable the Kerberos V5 functionality with the following commands:

```
$ MULTINET CONFIGURE /SERVER
... startup messages...
SERVER-CONFIG>SELECT TELNET
(The Selected SERVER entry is now TELNET)
```

```
SERVER-CONFIG>SET PROGRAM MULTINET:LOADABLE_KTELNET_CONTROL  
(Program to run for TELNET set to MULTINET:LOADABLE_KTELNET_CONTROL)  
SERVER-CONFIG>SET INIT Merge Image  
(Init action of TELNET set to Merge_Image)
```

After the values are saved and the Master Server is restarted, Kerberos 5 functionality is available.

The authentication behavior on the TELNET Server is determined by the system logical `MULTINET_TELNET_AUTH`. It has 3 possible values: `ALLOWED`, `REQUIRED`, and `DISABLED`.

**Note:** Not all configuration options are available with the Kerberos V5 Server. While the Kerberos V5 Server is started and terminated by the Master Server, it runs as a separate process. It uses a limited subset of server control options. Server control options currently supported are: `INIT`, `Program`, `Priority`, and `Log-Accepts`. To set the `SOCKET-PORT` option, use the system logical `MULTINET_TELNET_PORT`.

The default is `DISABLED`; a login prompt will result. When the value is `REQUIRED`, any user without a valid Kerberos V5 Ticket Granting Ticket (TGT) will be rejected. Finally, if the value is `ALLOWED`, the user can log-in to the server with or without a valid Kerberos V5 TGT (with a login prompt resulting if no TGT).

For example, to force authentication by any remote telnet client, set the logical as follows:

```
$ DEFINE/SYSTEM MULTINET_TELNET_AUTH REQUIRED
```

**Note:** Kerberos V5 requires Kerberos for OpenVMS (Version 2.0), which is available from the HP Web site. The Kerberos V5 applications can also run with any Kerberos V5-compliant Key Distribution Center (KDC) software. Kerberos V5 applies to VMS V7.0 or higher, and VMS Alpha V7.2-2 or higher.

## Configuring the TELNET Server for NTY Devices

The TELNET server parameters `INCLUDE-AUTHENTICATION-INDICATION`, `INCLUDE-PORT-NUMBER` and `INCLUDE-SEND-LOCATION` cause an authentication indication, a port number, or a send location (or all three) to be placed in the `TT_ACCPORNAM` field of the NTY device control block.

These parameters are disabled by default. To enable them, use `SERVER-CONFIG (MULTINET CONFIGURE /SERVER)`.

**Note:** These parameters will not function with the Kerberos 5 Telnet server, as it uses `pty` devices.

Enable these parameters with the following commands:

```
$ MULTINET CONFIGURE /SERVER
. . . startup messages . . .
SERVER-CONFIG>SELECT TELNET
[The Selected SERVER entry is now TELNET]
SERVER-CONFIG>SET PARAMETERS
You can now add new parameters for TELNET. An empty line terminates.
Add Parameter: INCLUDE-AUTHENTICATION-INDICATION
Add Parameter: INCLUDE-PORT-NUMBER
Add Parameter: INCLUDE-SEND-LOCATION
Add Parameter: RETURN
[Service specific parameters for TELNET changed]
SERVER-CONFIG>
```

If these parameters are defined, the appropriate information is stored in the `TT_ACCPORNAM` field.

These parameters appear in the Remote Port Info field of the `SHOW TERMINAL` and `SHOW USERS` command output. The `INCLUDE-SEND-LOCATION` parameter enables support for RFC 779 in the Telnet server. In the following example, the port number is 1021, and the `/AUTH` qualifier indicates that `WHORFIN` used authentication to log in from `LOT49.EXAMPLE.COM`.

```
$ SHOW TERMINAL
Terminal: _VTA84: Device_Type: VT100 Owner: WHORFIN
Physical terminal: _NTY3:
Remote Port Info: LOT49.EXAMPLE.COM/1021/AUTH
Input: 9600 LFill: 0 Width: 80 Parity: None
Output: 9600 CRfill: 0 Page: 24
Terminal Characteristics:
Interactive Echo Type_ahead No Escape
No Hostsync Ttsync Lowercase Tab
Wrap Scope Remote No Eightbit
No Broadcast No Readsyc No Form Fulldup
No Modem No Local_echo Autobaud Hangup
No Brdcstmbx No DMA Altypeahd Set_speed
No Commsync Line Editing Insert editing No Fallback
No Dialup No Secure server Disconnect No Psthru
No Syspassword No SIXEL Graphics No Soft Characters No Printer Port
```

```
Numeric Keypad ANSI_CRT No Regis No Block_mode
Advanced_video No Edit_mode DECEC_CRT2
No DEC_CRT3 No DEC_CRT4 VMS Style Input
$
```

Authentication information is not valid if someone uses the `LOGOUT /NOHANG` command and then logs in again.

## Configuring SYSLOG

SYSLOG receives messages from remote IP nodes that have been configured to forward SYSLOG messages to the MultiNet host. SYSLOG then directs the messages to a file, terminal, or OPCOM. In addition, messages can be forwarded elsewhere. (Forwarding messages are specified with the Forwarding command in the SYSLOG configuration file)

SYSLOG is provided with MultiNet, and you can enable it with `SERVER-CONFIG`. By default, when MultiNet is installed, SYSLOG is disabled.

**Note:** Messages generated by the `MULTINET_SERVER` process are not sent via SYSLOG, but are instead directed to OPCOM. OPCOM copies the messages to the `SYS$MANAGER:OPERATOR.LOG` file.

With SYSLOG enabled, you can determine the precise output of message classes and specify how priority messages are handled. Message classes and facility keywords are described below.

Message Type	Facility Keyword
Authorization messages	auth
BOOTP messages	bootpd
Daemon (background processes) messages	daemon
Domain Name System messages	named

GATED (gateway messages)	gated
Kernel messages	kern
LPR messages	lpr
mesages from local facilities	local0 - local7
NEWS messages	news
Mail utility messages	mail
Network Time Protocol (NTP) messages	ntpd
PPP messages	ppp
Security services messages	security
Messages generated by user programs	user

## Enabling SYSLOG

Enable SYSLOG with `SERVER-CONFIG` and modify the file `MULTINET:SYSLOG.CONFIGURATION`. Enable SYSLOG as follows:

```
$ MULTINET CONFIGURE /SERVER
MultiNet Server Configuration Utility 5.6(nnn)

[Reading in configuration from MULTINET:SERVICES.MASTER_SERVER]
SERVER-CONFIG>ENABLE SYSLOG
SERVER-CONFIG>EXIT
  [Writing configuration to MULTINET_COMMON_ROOT:[MULTINET]
SERVICES.MASTER_SERVER]
$
```

Add entries to the configuration file in the following form: `selector action`

Separate the fields with tabs. The selector is a semicolon-separated list of priority specifications in this form:

```
facility.level[;facility.level]
```

*facility* is a keyword (see the above table for a list of SYSLOG facility keywords).

Both *facility* and *level* are generated by applications on the remote host. The action specifies how SYSLOG responds to these messages. If the applications on the remote host do not write messages to SYSLOG, they are not displayed.

Possible values for *facility* are *auth*, *bootpd*, *daemon*, *gated*, *kern*, *mail*, *named*, *ntpd*, *security*, *user*, and asterisk (\*). (Specify an asterisk to include all messages written to SYSLOG of the specified priority.)

Each facility represents a different source of system message. The level is the priority level for each message. Possible values ranging from most severe to least severe are *panic*, *emerg*, *alert*, *crit*, *err*, *error*, *warn*, *warning*, *notice*, *info*, *debug*, and asterisk (\*). (Specify an asterisk to include all messages for the specified level.)

Examples of *facility.level* statements are:

Example	Description
*.debug	All debug messages
ntp.info	Network Time Protocol information messages
gated.warn	GATED warning messages

The *action* is the destination of the message. Possible message destinations include:

Destination	Specified As	Example
Broadcast (RWALL)	Asterisk	*
Console	Use OPCOM	/OPCOM
File	Precede the fully specified file name with a slash	/USERS:[HOLMES]OUTFILE.
Forwarding	Precede the IP address or host name with an at sign (@)	@192.92.38.1

OPCOM	Precede with a slash	/OPCOM
Terminal	Precede the terminal name with a slash and end name with a colon	/TXA3:

Each message handled by SYSLOG includes a time stamp, the facility name at the beginning of the message, and a newline character at the end of the message.

Comments are entered in the `SYSLOG.CONFIGURATION` file as pound signs (#) at the beginning of the line.

## SYSLOG Configuration File Examples

Some example `SYSLOG.CONFIGURATION` file entries are shown below:

```
# SYSLOG.CONFIGURATION examples
#
# Each entry is tab-separated and has this form:
#      selector      action
#
*.debug                /OPCOM
kern.panic;kern.warning /OPCOM
syslog.info            /users:[treefrog]syslog_info_messages.
*.error                @forwardhost
user.*                 /users:[treefrog]all_user_messages.
```

# 10. Configuring Network Routing

This chapter describes how to configure network routing.

## Routing Methods Overview

MultiNet provides two routing methods:

The IP routing table (also known as static routing)	Specifies the IP addresses of destinations and gateways for your communications. Use static routing when you have a very simple network, or a moderately complex network with a configuration that does not change often. Static routing is simple to configure and use.
The GATED dynamic gateway routing daemon	Handles multiple routing protocols and replaces Routed and EGPUP. GATED is used when node or gateway reconfigurations occur frequently, where the network geometry changes as networks come and go, or when dynamic routing protocols are being used for other reasons. GATED is more difficult to configure and use, but is much more flexible than static routing and provides a much greater selection of features.

## Configuring Static IP Routes

The `MULTINET SET /ROUTE` command specifies static IP routing, including the default route. This command is normally invoked automatically by the network startup command file generated by the MultiNet Network Configuration Utility (`NET-CONFIG`).

**Note:** If the GATED gateway routing service is enabled, it takes full control of your routing tables and removes any statically-defined routes that are not also specified in the GATED configuration.



## Adding Static Routes

You can configure a static default route using `NET-CONFIG`.

You can add additional static routes by creating the command file `MULTINET:LOCAL_ROUTES.COM`, which is executed as part of the network startup.

An example of the command for configuring a static route is:

```
$ MULTINET SET/ROUTE/ADD=(DEST=DEFAULT,GATE=192.0.0.1)
```

## Changing the Default Route

To change the default route, first delete the current default route, then add the new one:

```
$ MULTINET SET/ROUTE/DELETE=(DEST=DEFAULT,GATE=IP address)
$ MULTINET SET/ROUTE/ADD=(DEST=DEFAULT,GATE=IP address)
```

## Using GateD

The Gateway Routing Daemon (GateD) manages multiple routing protocols, including the Routing Information Protocol (RIP), Local Network Protocol (HELLO), Router Discovery Protocol, Open Shortest Path First (OSPF) protocol, Exterior Gateway Protocol (EGP), and Border Gateway Protocol (BGP).

Using GateD, the network administrator can control the flow of routing information through a configuration language. Once you start GateD, it makes routing decisions based on the data gathered by the routing protocols. If routing using GateD, use GateD exclusively.

GateD allows you to control importing and exporting routing information by:

- Individual protocol
- Source and destination Autonomous System (AS)
- Source and destination interface
- Previous hop router
- Specific destination address

You can assign preference levels for different combinations of imported routing information by using a flexible masking capability. In MultiNet, the name of the GateD process is GateD.

## GateD Configuration File

MultiNet stores GateD configuration information in the `MULTINET:GATED.CONF` file. You must create this file before you can use GateD. For details on GateD configuration, see *GateD Configuration Statements*.

## GateD Route Selection

GateD determines the "best" route using preference values set for each protocol or peer. Each route has a single associated preference value, even though you can set preferences at many places in the `GATED.CONF` file. The last (or most specific) preference value is the one GateD uses. Some protocols have a secondary preference, sometimes called a "tie-breaker."

The factors GateD uses in determining "best" routes include:

- The route with the numerically smallest `preference` value is preferred.
- For two routes with equal preferences, the route with the numerically smallest `preference2` (the "tie-breaker") is preferred.
- A route learned from an interior gateway protocol is preferred over a route learned from an exterior gateway protocol. Least preferred is a route learned indirectly by an interior protocol from an exterior protocol.
- If Autonomous System (AS) path information is available, it helps determine the most preferred route:
  - A route with an AS path is preferred over one without an AS path.
  - If the AS paths and origins are identical, the route with the lower metric is preferred.
  - A route with an AS path origin of interior protocol is preferred over one with an origin of exterior protocol. Least preferred is an AS path with an unknown origin.
  - A route with a shorter AS path is preferred.
- If both routes are from the same protocol and AS, the one with the lower metric is preferred.
- The route with the lowest numeric next-hop address is used.

Preference values range from 0 to 255. The below table summarizes the default preference values for routes learned in various ways.

Default preference value	Is defined by ... statement
--------------------------	-----------------------------

0	interface
10	ospf
20	gendefault (internally generated default)
30	redirect
40	kernel (routes learned using the socket route)
60	static
90	hello
100	rip
110	(point-to-point interfaces)
120	interfaces (routes to interfaces that are down)
130	aggregate/generate
150	ospf (AS external)
170	bgp
200	egp

## Starting and Stopping GateD

After creating the `MULTINET:GATED.CONF` file, you need to stop and restart GateD. Follow these steps:

1. Log in as the system manager.
2. Stop the GateD process by entering: `MULTINET GATED/STOP`

3. Restart the GateD process by entering: `@MULTINET : START_SERVER`

**Note:** It is not necessary to stop and restart GateD to get it to read a new MULTINET : GATED . CONF file, just use MULTINET GATED / LOAD.

## Configuring GATED

Use the commands in the table below to manage the GateD process. To use these commands, you need OPER or SYSPRV privilege.

Command	Description
MULTINET GATED / CHECK (file)	Checks a GateD configuration file for syntax errors
MULTINET GATED / DUMP	Dumps the state of the GateD process to a file
MULTINET GATED / LOAD	Loads a new GateD configuration file
MULTINET GATED / SET / TRACE	Controls tracing in GateD
MULTINET GATED / SHOW / OSPF / ADVERTISE	Shows OSPF link state advertisements
MULTINET GATED / SHOW / OSPF / AS	Shows the AS external database entries
MULTINET GATED / SHOW / OSPF / DESTINATIONS	Shows the list of destinations and their indices
MULTINET GATED / SHOW / OSPF / ERRORS	Shows the OSPF error log
MULTINET GATED / SHOW / OSPF / HOPS	Shows the set of next hops for the OSPF router queried
MULTINET GATED / SHOW / OSPF / INTERFACES	Shows all configured interfaces for OSPF

MULTINET GATED/SHOW/OSPF/LOG	Shows the cumulative OSPF log of input/output statistics
MULTINET GATED/SHOW/OSPF/ NEIGHBORS	Shows all OSPF routing neighbors
MULTINET GATED/SHOW/OSPF/ROUTING	Shows the OSPF routing table
MULTINET GATED/SHOW/OSPF/STATE	Shows the link state database (except AS Externals)
MULTINET GATED/SHOW/RIP	Queries Routing Information Protocol (RIP) gateways
MULTINET GATED/SHOW/TRACE	Shows tracing in GateD
MULTINET GATED/STOP	Stops the GateD process
MULTINET GATED/TOGGLE_TRACING	Toggles tracing in GateD
MULTINET GATED/UPDATE_INTERFACES	Rescans the GateD network interfaces

## GateD Configuration Statements

The GateD configuration file is `MULTINET:GATED.CONF`. This file must be present for the GateD process to run. The structure of the GateD configuration language is similar to C. The configuration file consists of statements terminated by a semicolon (;). Statements consist of tokens separated by a space. This structure simplifies identification of the associated parts of the configuration.

You can include comment lines either by beginning them with a pound sign (#) or delimiting them with slash asterisk (/\*) and asterisk slash (\*). The configuration file consists of the following sections, which reflect the order in which the statements, if used, must appear:

<b>Directives</b>	(%directory, %include)
-------------------	------------------------

<b>Statements</b>	traceoptions options interfaces
<b>Definitions</b>	autonomous-system routeid martians
<b>Protocols</b>	rip help redirect router-discovery server/client bgp ospf
<b>Static routes</b>	static
<b>Control</b>	import export aggregate generate

## Directives

Directive statements include:

```
%directory
%include
```

Directive statements provide special instructions to the parser. They do not relate to the protocol configuration and can occur anywhere in `GATED.CONF`. They also end in a new line instead of a semicolon (;) like the other statements.

# Format

## **%directory "directory"**

Defines the directory where the include files go if you do not fully specify directory as part of the filename in the %include statement. Does not actually change the current directory, but simply applies the directory prefix.

## **%include "filename"**

Identifies an include file. GateD includes the contents of the file in GATED.CONF at the point where the %include appears. If you do not fully specify the filename, it is relative to the directory defined in %directory. The %include directive causes GateD to parse the specified file completely before resuming. You can nest up to ten levels of include files.

---

# traceoptions

The `traceoptions` statement controls tracing options. You can configure GateD tracing options at many levels. These include file specifications, control options, and global and protocol-specific tracing options.

Lower levels of statements inherit tracing options from the next higher level, unless overridden.

## Format

```
traceoptions [ "tracefile" [replace] [size size[k | m] files files]]  
              [nostamp] traceoptions [except traceoptions] | none ;
```

## Options and Parameters

### **"*tracefile*"**

File to receive tracing information. If this filename is not fully specified, GateD creates it in the directory where you started GateD.

### **replace**

Replaces an existing file. The default is to append to an existing file.

### **size *size*[k | m] files *files***

Limits the maximum size, in k or m or the files indicated, of the trace file (the minimum is 10k). When the file reaches `size`, GateD creates a new version.

### **nostamp**

Control option which means not to prepend a timestamp to all trace lines. The default is to prepend a timestamp.

### ***traceoptions***

Specific to each protocol statement. Note that these global options may not apply to all protocols.

### **except *traceoptions***



Disables more specific trace options after enabling broader ones.

**none**

Turns off all tracing for the protocol or peer.

Option	Description
adv	For debugging: traces the allocation and freeing of policy blocks.
all	Turns on the <code>general</code> , <code>normal</code> , <code>policy</code> , <code>route</code> , <code>state</code> , <code>task</code> , and <code>timer</code> options.
general	Shorthand for specifying both the <code>normal</code> and <code>route</code> options.
iflist	Traces reading of the kernel interface. Useful to specify this with the <code>-t</code> option on the command line since the first interface scan occurs before reading the configuration file.
normal	Traces normal protocol occurrences (abnormal protocol occurrences are always traced).
parse	For debugging: traces the lexical analyzer and parser.
policy	Traces how protocol and user-specified policy apply to routes imported and exported.
route	Traces routing table changes for routes installed by the protocol or peer.
state	Traces state machine transitions in the protocols.
symbols	Traces symbols read from the kernel at startup. The only useful way to specify this level of tracing is to use the <code>-t</code> option on the command line, since GateD reads the symbols from the kernel before parsing the configuration file.
task	Traces system interface and processing associated with the protocol or peer.
timer	Traces timer usage by the protocol or peer.

---

# options

The options statements let you specify some global options. If used, options must appear before any other type of configuration statement in `GATED.CONF`.

## Format

```
options [nosend]
        [noresolve]
        [gendefault [preference value][gateway host] ]
        [syslog [upto] loglevel]
        [mark time] ;
```

## Options and Parameters

### **nosend**

Does not send packets. Makes it possible to run GateD on a live network to test protocol interactions, without actually participating in the routing protocols. You can examine the packet traces in the GateD log to verify that GateD functions properly. Most useful for RIP and HELLO. Does not yet apply to BGP, and not useful with EGP and OSPF.

### **noresolve**

Does not resolve symbolic names into IP addresses. By default, GateD uses the `gethostbyname()` and `getnetbyname()` library calls that usually use the Domain Name System (DNS) instead of the host's local host and network tables. If there is insufficient routing information to send DNS queries, GateD deadlocks during startup. Use this option to prevent these calls.

**Note:** When you use this option, symbolic names cause configuration file errors.

```
gendefault [preference value] [gateway host]  
nogendefault
```

Creates a default route with the special protocol default when a BGP or EGP neighbor is up. You can disable this for each BGP/EGP group with the `nogendefault` option. By default, this route has a preference value of 20. This route is normally not installed in the kernel forwarding table; it is only

present for announcement to other protocols. The `gateway` option installs the default route in the kernel forwarding table with a next hop of the gateway defined.

**Note:** Using more general options is preferred to using `gendefault`. (See *aggregate* for details on the `generate` statement.)

### **syslog [upto] loglevel**

Amount of data GateD logs to OPCOM. OpenVMS systems map UNIX `syslog` logging levels to OPCOM severity levels. The default is `syslog upto info`. The mapping of `syslog` to OPCOM logging levels appears in *Mapping of UNIX syslog Levels to OpenVMS OPCOM Severity Levels*.

### **mark time**

GateD sends a message to the trace log at the specified `time` interval. Can be one method of determining if GateD is still running.

<b>syslog log level</b>	<b>Is equivalent to OPCOM level...</b>
emerg	FATAL
alert	FATAL
crit	FATAL
err	ERROR
warning	WARNING
notice	INFORMATIONAL
info (default)	INFORMATIONAL
debug	INFORMATIONAL

## Example

```
# generate a default route when peering with an EGP or BGP neighbor:  
#  
options gendefault ;
```

---

# interfaces

An interface is the connection between a router and one of its attached networks. Specify a physical interface by interface name, IP address, or domain name. Multiple reference levels in the configuration language let you identify interfaces using wildcards (only the device driver part of the name, to match any unit number), interface type names, or addresses.

## Format

```
Interfaces
{
  options
    [strictinterfaces]
    [scaninterval time] ;
  interface list
    [preference value]
    [down preference value]
    [passive]
    [simplex]
  define address
    [broadcast address] | [pointtopoint address]
    [netmask mask]
    [multicast] ;
};
```

## Options Clause

```
options
    [strictinterfaces]
    [scaninterval time] ;
```

### **strictinterfaces**

Makes it a fatal error to use reference interfaces not present when you start GateD or that are not part of the `define` parameter. Normally, GateD issues a warning message and continues.

### **scaninterval *time***

Sets how often GateD scans the kernel interface list for changes. The default is every 15 seconds on most systems, and 60 seconds on systems that pass interface status changes through the routing socket (such as BSD 4.4).

## Interface Clause

Sets interface options on the specified interfaces. A *list* can consist of interface names, domain names, numeric addresses, or the value `all`. Include one or more interface names, including wildcard names (without a number) and those that can specify more than one interface or address.

There are three ways to reference an interface:

<b>By wildcard</b>	Only the device driver part of the name, to match any unit number.
<b>By name</b>	Combined device driver and unit number of an interface.
<b>By address</b>	IP address or domain name (if resolving to one address only).

There are four types of interfaces allowed:

<b>Loopback</b>	Must have the address 127.0.0.1. Packets from a loopback interface go back to the originator. Also used for reject and blackhole routes ( <b>not supported in MultiNet</b> ). The interface ignores any net mask. It is useful to assign an additional address to the loopback interface that is the same as the OSPF or BGP router ID; this allows routing to a system based on router ID that works if some interfaces are down.
<b>Broadcast</b>	Multiaccess interface capable of physical level broadcast, such as Ethernet, Token-Ring, and FDDI. A broadcast interface has an associated subnet mask and broadcast address. The interface route to a broadcast network is a route to the complete subnet.
<b>Point-to-point</b>	Tunnel to another host, usually on some sort of serial link. A point-to-point interface has a local address and a remote address. The remote address must be unique among the interface addresses on a given router. Many point-to-point interfaces and up to one non point-to-point interface must share the local address. This conserves subnets as you do not need any when using this technique. If you use a subnet mask on a point-to-point interface, only RIP version 1 and HELLO use it to determine which subnets propagate to the router on the other side of the point-to-point interface.
<b>Non-broadcast multiaccess (NBMA)</b>	Multiaccess but not capable of broadcast, such as frame relay and X.25. This type of interface has a local address and a subnet mask.

**preference value**

Sets the preference for routes to this interface when it is up and GateD determines it to function properly. The default preference value is 0. While the preference statement is optional, it is strongly recommended that you set an explicit preference value if you do use it.

**down preference value**

Sets the preference for routes to this interface when GateD determines that it does not function properly, but the kernel does not indicate that it is down. The default down preference value is 120.

**passive**

Does not change the preference of the route to the interface if determined not to function properly from lack of routing information. GateD checks this only if the interface actively participates in a routing protocol.

**simplex**

The interface does not recognize its own broadcast packets. Some systems define an interface as simplex with the IFF\_SIMPLEX flag. On others, the configuration defines it. On simplex interfaces, packets from the local host are assumed to have been looped back in software and are not used to indicate that the interface functions properly.

---



# Define Clause

## Interfaces

```
{
  define address
    [broadcast address] | [pointtopoint address]
    [netmask mask]
    [multicast] ;
} ;
```

Defines interfaces not present when starting GateD so that the configuration file can reference them when using options `strictinterfaces`.

### **broadcast address**

Makes the interface broadcast-capable (for Ethernet or Token-Ring) and specifies the broadcast address.

### **pointtopoint address**

Makes the interface point-to-point (such as SLIP or PPP) and specifies the address on the local side of the interface. The first address in the `define` statement references the host on the remote end of the interface.

An interface not defined as `broadcast` or `pointtopoint` must be `nonbroadcast multiaccess` (NBMA), such as for an X.25 network.

### **netmask mask**

Subnet mask to use on the interface. Ignored on point-to-point interfaces.

### **multicast**

Makes the interface multicast-capable.

## Examples

1. This example sets the interface as passive.

```
# do not mark interface 192.168.95.41 as down,
# even if there is no traffic:
#
interfaces{
```

```
interface 192.168.95.41 passive ;
} ;
```

2. This example shows the interface statements used with the `rip` statement (see the `rip` description). Users would receive RIP packets only from interfaces `sva-0` and `sva-1`, but not from `fza-0`, and `sva-1` would be the only one that could send them.

```
rip yes {
  interface all noripin noripout ;
  interface sva ripin
;
  interface sva-1 ripout ;
} ;
```

## Definition Statements

Definition statements include:

```
autonomoussystem
routerid
martians
```

Definition statements are general configuration statements that relate to all of GateD or at least to more than one protocol. You must use these statements for any protocol statements in the configuration file.

### Format

```
autonomoussystem ASnumber [loops number];
```

An autonomous system (AS) is a set of routers under a single technical administration, using an internal protocol and common metrics to route packets within the AS, and an external protocol to route packets to other ASs. The Network Information Center (NIC) assigns AS numbers.

The `autonomoussystem` statement sets the AS number of the router. You require this option if using BGP or EGP. The `loops` option is only for protocols supporting AS paths, such as BGP. It controls the number of times this AS can appear in an AS path, and defaults to 1.

```
routerid host ;
```

A router ID is an IP address used as a unique identifier assigned to represent a specific router, usually the address of an attached interface. The `routerid` statement sets the router ID for the BGP and OSPF

protocols. The default is the address of the first interface GateD encounters. The address of a non-point-to-point interface is preferred over the local address of a point-to-point interface, and an address on a loopback interface that is not the loopback address (127.0.0.1) is most preferred.

```
martians
{
  host host [allow] ;
  network [allow] ;
  network mask mask [allow] ;
  network masklen number [allow] ;
  default [allow] ;
} ;
```

The `martians` statement defines a list of invalid addresses, called *martians*, that the routing software ignores. Sometimes a misconfigured system sends out obviously invalid destination addresses. The statement allows additions to the list of `martian` addresses. (See *Route Filtering* for details on specifying ranges.)

You can also use the `allow` parameter to explicitly allow a subset of an otherwise disallowed range.

## Example

This example shows the use of all three definition statements, `autonomousssystem`, `routerid`, and `martians`.

```
# use AS number 249:
#
autonomousssystem 249 ;
#
# set the router
ID number:
#
routerid 192.168.95.41 ;
#
# prevent routes to
0.0.0.26 from ever being accepted:
#
martians {
host 0.0.0.26 ;
};
```

---



# Route Filtering

You can filter routes by matching a certain set of routes by destination, or by destination and mask. Use route filters on `martians`, `import`, and `export` statements.

The action taken when no match is found depends on the context. For example, `import` and `export` route filters assume an `all reject` ; at the end of a list. A route matches the most specific filter that applies. Specifying more than one filter with the same destination, mask, and modifiers generates an error.

## Format

```
network [exact | refines | allow]
network mask mask [exact | refines]
network masklen number [exact | refines]
all
default
host host
```

## Options and Parameters

### **network**

Destination network IP address. You can use one of the following options:

<code>exact</code>	Destination mask must match the supplied mask exactly. Used to match a network, but no subnets or hosts of that network.
<code>refines</code>	Destination mask must be more specified (longer) than the filter mask. Used to match subnets or hosts of a network, but not the network.
<code>allow</code>	See the <code>martians</code> definition statement.

### **mask mask**

Destination network mask.

### **masklen number**

Length of the destination network mask.

**all**

Entry matches anything. Equivalent to `0.0.0.0 mask 0.0.0.0`.

**default**

Matches the default route. To match, the address must be the default address and the mask must be all zeros. Equivalent to `0.0.0.0 mask 0.0.0.0 exact`. (Not valid for `martians` statements.)

**host *host***

Matches the specific host. To match, the address must match exactly the specified host, and the network mask must be a host mask (all 1s). Equivalent to `host mask 255.255.255 exact`. (Not valid for `martians` statements.)

---

# rip

GateD supports the Routing Information Protocol (RIP). RIP is a distance-vector protocol for distributing routing information at the local network level of the Internet. In distance-vector routing, each router transmits destination addresses and costs to its neighbors (computers communicating over RIP).

RIP versions 1 and 2 are the most commonly used interior protocol. RIP selects the route with the lowest metric as the best route. The metric is a hop count representing the number of gateways through which data must pass to reach its destination. The longest path that RIP accepts is 15 hops. If the metric is greater than 15, a destination is considered unreachable and GateD discards the route. RIP assumes the best route uses the fewest gateways, that is, the shortest path, not taking into account congestion or delay along the way.

RIP uses two types of packets: requests and responses.

**Requests.** A request asks for information about specific destinations or for all destinations. RIP can send requests when a router:

- Comes up
- Receives timed-out information about a destination

If a request fails to specify a destination, RIP assumes the router requests information about all destinations.

**Responses.** Responses contain destination and cost pairs. RIP sends responses under the following three conditions:

- In response to a request
- When information changes; for example, cost information
- At set intervals; for example, reporting the destination to each neighbor every 30 seconds

RIP discards the destination and cost information if a neighbor fails to report the distance to a destination after a certain time interval.

**RIP IP Addresses.** RIP version 1 contains no provision for passing around a mask. RIP infers the mask based on whether the address is class A, B, or C. Sometimes there are special cases when the inferred mask differs from class A, B, or C. For example:

- When you use RIP with a subnet (in this case the routers must know the subnet mask for a particular network number)
- When the system updates RIP with an address reported as 0.0.0.0, RIP considers this address as a default destination with a mask of 0.0.0.0

- When the system updates RIP with bits set in the host portion of the address, RIP assumes the address refers to a host with a mask of 255.255.255.255

With RIP version 2, you can specify the network mask with each network in a packet.

**Configuring RIP.** You configure RIP in the `GATED.CONF` file using a GateD protocol statement that enables or disables RIP. The syntax of the `rip` statement is as follows, with the parameters described next:

## Format

```
rip yes | no | on | off
[ {
    [no]broadcast ;
    nocheckzero ;
    preference value ;
    defaultmetric metric ;
    query authentication [ none | [ [simple | md5] password ] ];
    interface list
        [[no]ripin ] [ [no]ripout ]
        [metricin metric]
        [metricout metric] ;
        [version 1] | [ version 2 [multicast | broadcast] ]
        [ [secondary] authentication [none | [[simple | md5] password ]]];
    trustedgateways list ;
    sourcegateways list ;
    traceoptions options ;
} ] ;
```

## Options and Parameters

**yes | on (default)**

**no | off**

When enabled on a host, RIP listens in the background to routing updates. When enabled on a gateway, RIP supplies routing updates. Enabled by default.

**broadcast ;**

Broadcasts RIP packets regardless of the number of interfaces present. Useful when propagating static routes or routes learned from another protocol into RIP. In some cases, using **broadcast** when only one network interface is present can cause data packets to traverse a single network twice. The default for more than one interface.



**nobroadcast ;**

Does not broadcast RIP packets on attached interfaces even if there is more than one. If you use the `sourcegateways` parameter, routes are still unicast directly to that gateway. The default for a single interface.

**nocheckzero ;**

Does not make sure that reserved fields in incoming RIP version 1 packets are zero. Normally RIP rejects packets whose reserved fields are zero.

**preference value ;**

Sets the preference for routes learned from RIP. A preference specified in import policy can override this. The default `preference value` is 100.

**defaultmetric metric ;**

Metric used when advertising routes learned from other protocols. Choice of values requires that you explicitly specify a metric in order to export routes from other protocols into RIP. A metric specified in export policy can override this. The default `metric` is 16.

**query authentication ;**

Authentication required of query packets that do not originate from routers. The default is `none`.

---

# Interface Clause

```
rip yes | no | on | off
[ {
    [no]broadcast ;
    nocheckzero ;
    preference value ;
    defaultmetric metric ;
    query authentication [ none | [ [simple | md5] password ] ] ;
    interface list
        [ [no]ripin ] [ [no]ripout ]
        [metricin metric]
        [metricout metric] ;
        [version 1] | [ version 2 [multicast | broadcast] ]
        [ [secondary] authentication [none | [ [simple | md5] password] ] ] ;
        trustedgateways list ;
        sourcegateways list ;
        traceoptions options ;
    } ] ;
```

Controls various attributes of sending RIP on specific interfaces. (See the interfaces statement for a description of `list`.) Note that if there are multiple interfaces configured on the same subnet, only the first one on which RIP output is configured sends the RIP updates. This limitation is required because of the way the UNIX kernel operates. A future GateD release will hopefully remove this limitation. The default `list` value is `all`.

**ripin** (default)

**noripin**

Use `ripin` explicitly when using `noripin` on a wildcard interface descriptor. The `noripin` option ignores RIP packets received over the specified interfaces.

**ripout** (default)

**noripout**

Use `ripin` explicitly when using `noripout` on a wildcard interface descriptor. The `noripin` does not send RIP packets over the specified interfaces.

**metricin metric**

RIP metric to add to incoming routes before they are installed in the routing table. Makes the router prefer RIP routes learned using the specified interfaces less than those learned from other interfaces. The default is the kernel interface metric plus 1. If using this as the absolute value, the kernel metric is not added.

**metricout *metric***

RIP metric to add to routes sent over the specified interface(s). Makes other routers prefer other sources of RIP routes over this router. The default `metric` value is 0.

**version 1** (default)

Sends RIP version 1 packets over the specified interface(s).

**version 2 [multicast | broadcast]**

Sends RIP version 2 packets over the specified interfaces. If IP multicasting support is available on this interface, the default is to send full version 2 packets. If multicasting is not available, version 1 compatible version 2 packets are sent. Options include:

multicast	Multicasts RIP version 2 packets over this interface. (Default)
broadcast	Broadcasts RIP version 1 compatible version 2 packets over this interface even if IP multicasting is available

**[secondary] authentication [none | [ [simple | md5] password] ]**

Authentication type to use. Applies only to RIP version 2 and is ignored for RIP-1 packets. If you specify a *password*, the authentication type defaults to `simple`. The password should be a quoted string with 0 to 16 characters. If you specify `secondary`, this defines the secondary authentication. The default is `authentication none`.

**trustedgateways *list***

List of gateways from which RIP accepts updates (host names or IP addresses). If used, only updates from the gateways in the list are accepted. The default `list` value is `all`.

**sourcegateways *list***

List of routers to which RIP sends packets directly, not through multicasting or broadcasting. If used, only updates from the gateways in the list are accepted. The default `list` value is `all`.

**traceoptions *options***

RIP-specific trace options:

packets	All RIP packets, or packets [detail] send or [detail] recv (detail provides a more verbose format to provide more details; if used, detail must come before send or recv)
request	RIP information request packets, such as REQUEST, POLL and POLLENTRY
response	RIP RESPONSE packets that actually contain routing information

---

# hello

GateD supports the HELLO protocol. HELLO is an interior protocol that uses delay as the deciding factor when selecting the best route. Delay is the round trip time between source and destination. HELLO is not as widely used as when it was the interior protocol of the original 56-Kb/sec NSFNET backbone and used between LSI-11 ("fuzzball") routers. Because of this, HELLO is disabled by default.

By default, HELLO, like RIP, uses the kernel interface metric set by the `ifconfig` command to influence metrics added to routes as they are installed in the routing table (`metricin`). Since the kernel interface metric is in hops, it must be translated into HELLO's millisecond metric. For the translation scheme, see the HELLO Hops-to-Metrics Translation table below.

<b>This many Hops</b>	<b>Translate to this HELLO metric</b>	<b>This many Hops</b>	<b>Translate to this HELLO metric</b>	<b>This many Hops</b>	<b>Translate to this HELLO metric</b>
0	0	6	713	12	75522
1	100	7	1057	13	11190
2	148	8	1567	14	16579
3	219	9	2322	15	24564
4	325	10	3440	16	3000
5	481	11	5097		

You configure HELLO in the GATED.CONF file using a GateD protocol statement that enables or disables HELLO.

When enabled, HELLO assumes `nobroadcast` when only one interface exists. HELLO assumes broadcast when more than one interface exists.

## Format

```
hello yes | no | on | off
[ {
    [no]broadcast ;
    preference value ;
    defaultmetric metric ;
    interface list
        [ [no]helloin ]
        [ [no]helloout ]
        [metricin metric]
        [metricout metric] ;
    trustedgateways list ;
    sourcegateways list ;
    traceoptions options ;
} ] ;
```

## Options and Parameters

### **yes | on or no | off (default)**

When enabled on a host, HELLO listens in the background for routing updates. When enabled on a gateway, HELLO supplies routing updates. Disabled by default.

### **broadcast ;**

### **nobroadcast ;**

The `broadcast` option broadcasts HELLO packets regardless of the number of interfaces present. Useful when propagating static routes or routes learned from another protocol into HELLO. In some cases, using `broadcast` when only one network interface is present can cause data packets to traverse a single network twice. The default for more than one interface.

The `nobroadcast` option does not broadcast HELLO packets on attached interfaces, even if there is more than one. If you use the `sourcegateways` parameter, routes are still unicast directly to that gateway. The default for a single interface.

### **preference value ;**

Preference for routes learned from HELLO. A preference specified in import policy can override this. The default preference value is 90.

### **defaultmetric metric ;**

Metric used when advertising routes learned from other protocols. Requires you to explicitly specify a metric in order to export routes from other protocols into HELLO. A metric specified in export policy can override this. The default `metric` is 30000.

```
interface list  
    [ [no]helloin ]  
    [ [no]helloout ]  
    [metricin metric]  
    [metricout metric] ;
```

Controls various attributes of sending HELLO on specific interfaces. (See `interfaces` statement for a description of `list`.) Note that if there are multiple interfaces configured on the same subnet, only the first interface that has HELLO output configured sends the HELLO updates. This limitation is required because of the way the UNIX kernel operates. A future GateD release will hopefully remove this limitation. The default interface `list` value is `all`.

**helloin** (default)

**nohelloin**

Use `helloin` explicitly when using `nohelloin` on a wildcard interface descriptor. The `nohelloin` option ignores HELLO packets received over the specified interfaces.

**helloout** (default)

**nohelloout**

Use `helloout` explicitly when using `nohelloout` on a wildcard interface descriptor. The `nohelloout` option does not send HELLO packets over the specified interfaces.

**metricin metric**

HELLO metric to add to incoming routes before GateD installs them in the routing table. Makes this router prefer HELLO routes learned from other interfaces over those from the specified interface(s). The default is the kernel interface metric plus one. If using this as the absolute value, GateD does not add the kernel metric to the routing table.

**metricout metric**

HELLO metric to add to routes that are sent over the specified interface(s). Makes other routers prefer other sources of HELLO routes over this router. The default `metric out metric` value is 0.

**trustedgateways list**

List of gateways from which HELLO accepts updates (host names or IP addresses). If used, HELLO accepts only updates from the gateways in the list. The default `list` value is `all`.

**sourcegateways list**

List of routers to which HELLO sends packets directly, not through multicasting or broadcasting. If used, HELLO accepts only updates from the gateways in the list. The default `list` value is `all`.

**traceoptions packets**

All HELLO packets, or packets `[detail] send` or `[detail] rcv` (`detail` provides a more verbose format to provide more details; if used, `detail` must come before `send` or `rcv`).

---



# icmp

On systems without the BSD routing socket, GateD listens to ICMP messages received by the system. Processing of ICMP redirect messages is handled by the `redirect` statement.

Currently the only reason to specify the `icmp` statement is to be able to trace the ICMP messages that GateD receives.

## Format

```
icmp { traceoptions options ; }
```

## Options and Parameters

```
traceoptions options ;
```

ICMP tracing options (which you can modify with `detail` and `recv`) are as follows:

<code>packets</code>	All ICMP packets received
<code>redirect</code>	Only ICMP Redirect packets received
<code>routerdiscovery</code>	Only ICMP Router Discovery packets received
<code>info</code>	Only ICMP informational packets, which include mask request/response, info request/response, echo request/response and timestamp request/response
<code>error</code>	Only ICMP error packets, which include time exceeded, parameter problem, unreachable and source quench

---

# redirect

GateD controls whether ICMP redirect messages can modify the kernel routing table. If disabled, GateD only prevents a system from listening to ICMP redirects. By default, ICMP redirects are enabled on hosts, and disabled on gateways that run as RIP or HELLO suppliers.

You configure ICMP redirect handling in the `GATED.CONF` file using a GateD protocol statement.

## Format

```
redirect yes | no | on | off
{{
    preference value ;
    interface list [ [no]redirects ] ;
    trustedgateways list ;
}};
```

## Options and Parameters

**yes | on**

**no | off**

Enabled by default on hosts. Disabled by default on gateways running as RIP or HELLO suppliers.

### **preference value**

Preference for routes learned from a redirect. The default preference value is 30.

### **interface list [ [no]redirects ]**

Enables and disables redirects interface by interface. (See *interfaces* for a description of *list*.) The default interface *list* value is `all`. The possible parameters are:

<code>redirects</code>	May be necessary when you use <code>noredirects</code> on a wildcard interface descriptor. (Default)
<code>noredirects</code>	Ignores redirects received over the specified interface(s). The default is to accept redirects on all interfaces.

### **trustedgateways list**

List of gateways from which redirects are accepted (host names or addresses). By default, all routers on the shared network(s) are trusted to supply redirects. If used, only redirects from the gateways in the list are accepted. The default `list` value is `all`.

---

## routerdiscovery server

The Router Discovery Protocol is an IETF standard protocol used to inform hosts of the existence of routers without having hosts wiretap routing protocols such as RIP. Use it in place of, or in addition to, statically configured default routes in hosts.

The protocol is in two parts, the server that runs on routers and the client that runs on hosts (see the next statement). GateD treats these much like two separate protocols that you can enable only one at a time.

The Router Discovery Server runs on routers and announces their existence to hosts. It does this by periodically multicasting or broadcasting a Router Advertisement to each interface on which it is enabled. These Router Advertisements contain a list of all router addresses on a given interface and their preference for use as a default router.

Initially these Router Advertisements occur every few seconds, then fall back to occurring every few minutes. In addition, a host may send a Router Solicitation to which the router will respond with a unicast Router Advertisement (unless a multicast or broadcast advertisement is due momentarily).

Each Router Advertisement contains an Advertisement Lifetime field indicating how long the advertised addresses are valid. This lifetime is configured such that another Router Advertisement is sent before the lifetime expires. A lifetime of zero indicates that one or more addresses are no longer valid.

On systems supporting IP multicasting, the Router Advertisements are sent to the all-hosts multicast address 224.0.0.1 by default. However, you can specify `broadcast`. When Router Advertisements are being sent to the all-hosts multicast address, or an interface is configured for the limited-broadcast address 255.255.255.255, all IP addresses configured on the physical interface are included in the Router Advertisement. When the Router advertisements are being sent to a net or subnet broadcast, only the address associated with that net or subnet is included.

**Note:** Do not mix `routerdiscovery server` and `routerdiscovery client` statements in the `GATED.CONF` file or you may get unintended results. You should also include preference statements in the interfaces and `routerdiscovery` statements whenever possible.

### Format

```
routerdiscovery server yes | no | on | off
{{
```

```
traceoptions state ;
interface list
  [minadvinterval time]
  [maxadvinterval time]
  [lifetime time] ;
address list
  [advertise] | [ignore]
  [broadcast] | [multicast]
  [ineligible] | [preference value] ;
}};
```

**Note:** Interface *must* be mentioned in the “Interface” directive.

## Options and Parameters

**yes** | **on**

**no** | **off**

Enables or disables Router Discovery Protocol Server.

### **traceoptions state**

The **state** is the only trace option, which traces the state transitions. The Router Discovery Server does not directly support packet tracing options; tracing of router discovery packets is enabled through the `icmp` statement described in the *icmp* statement section.

### **interface list**

Parameters that apply to physical interfaces. Note a slight difference in convention from the rest of GateD: `interface` specifies just physical interfaces, while `address` specifies protocol (in this case, IP) addresses.

### **maxadvinterval time**

Maximum time allowed between sending broadcast or multicast Router Advertisements from the interface. Must be no less than 4 and no more than 30:00 (30 minutes). The default is 10:00 (10 minutes).

### **minadvinterval time**

Minimum time allowed between sending unsolicited broadcast or multicast Router Advertisements from the interface. Must be no less than 3 seconds and no greater than `maxadvinterval`. The default is `0.75 X maxadvinterval`.

**lifetime time**

Lifetime of addresses in a Router Advertisement. Must be no less than `maxadvinterval` and no greater than `2:30:00` (two hours, thirty minutes). The default is `3 X maxadvinterval`.

**address list**

Parameters that apply to the specified set of addresses on this physical interface. Note a slight difference in convention from the rest of GateD: `interface` specifies just physical interfaces while `address` is protocol (in this case, IP) addresses.

**advertise (default)**

**ignore**

The `advertise` keyword includes the specified addresses in Router Advertisements. The `ignore` keyword does not.

**broadcast**

**multicast**

The `broadcast` keyword includes the given addresses in a broadcast Router Advertisement because this system does not support IP multicasting, or some hosts on an attached network do not support IP multicasting. It is possible to mix addresses on a physical interface such that some are included in a broadcast Router Advertisement and some are included in a multicast Router Advertisement. This is the default if the router does not support IP multicasting.

The `multicast` keyword includes the given addresses in a multicast Router Advertisement. If the system does not support IP multicasting, the address(es) is not included. If the system supports IP multicasting, the default is to include the addresses in a multicast Router Advertisement if the given interface supports IP multicasting. If not, the addresses are included in a broadcast Router Advertisement.

**preference value**

**ineligible**

The `preference` keyword sets the preferability of the addresses as a default router address, relative to other router addresses on the same subnet. A 32-bit, signed, two's complement integer, with higher values meaning more preferable. Note that hex 80000000 may only be specified as ineligible. The default value is 0. Use a `preference` statement whenever possible.

The `ineligible` keyword assigns the given addresses a preference of hex 80000000, which means that it is not eligible to be the default route for any hosts. This is useful when the addresses should not be used as a default route, but are given as the next hop in an ICMP Redirect. This allows the hosts to verify that the given addresses are up and available.

---

# routerdiscovery client

A host listens for Router Advertisements through the all-hosts multicast address (224.0.0.2) if IP multicasting is available and enabled, or on the interface's broadcast address. When starting up, or when reconfigured, a host may send a few Router Solicitations to the all-routers multicast address, 224.0.0.2, or the interface's broadcast address.

When a Router Advertisement with a non-zero lifetime is received, the host installs a default route to each of the advertised addresses. If the preference is ineligible, or the address is not on an attached interface, the route is marked unusable but retained. If the preference is usable, the metric is set as a function of the preference such that the route with the best preference is used. If more than one address with the same preference is received, the one with the lowest IP address will be used. These default routes are not exportable to other protocols.

When a Router Advertisement with a zero lifetime is received, the host deletes all routes with next hop addresses learned from that router. In addition, any routes learned from ICMP Redirects pointing to these addresses will be deleted. The same happens when a Router Advertisement is not received to refresh these routes before the lifetime expires.

**Note:** Do not mix `routerdiscovery server` and `routerdiscovery client` statements in the `GATED.CONF` file or you may get unintended results. You should also include preference statements in the interfaces and `routerdiscovery` statements whenever possible.

## Format

```
routerdiscovery client yes | no | on | off
[
  traceoptions state ;
  preference value ;
  interface list
    [enable] | [disable]
    [broadcast] | [multicast]
    [quiet] | [solicit] ;
}] ;
```

## Options and Parameters

`yes` | `on`



**no | off**

Enables or disables the Router Discovery Protocol Client.

**traceoptions state ;**

The `state` is the only trace option, which traces the state transitions. The Router Discovery Server does not directly support packet tracing options; tracing of router discovery packets is enabled through the `icmp` statement described in the `icmp` statement section.

**preference value ;**

Preference of all Router Discovery default routes. Use a `preference` statement whenever possible. Default is 55.

**interface list**

Parameters that apply to physical interfaces. Note a slight difference in convention from the rest of GateD: `interface` specifies just physical interfaces. The Router Discovery Client has no parameters that apply only to interface addresses.

**enable** (default)

**disable**

Either performs or does not perform Router Discovery on the specified interfaces.

**broadcast**

**multicast**

The `broadcast` keyword broadcasts Router Solicitations on the specified interfaces. This is the default if IP multicast support is not available on this host or interface.

The `multicast` keyword multicasts Router Solicitations on the specified interfaces. If IP multicast is not available on this host and interface, no solicitation is performed. The default is to multicast Router Solicitations if the host and interface support it, otherwise Router Solicitations are broadcast.

**solicit** (default)

**quiet**

Either sends or does not send Router Solicitations on this interface, even though Router Discovery is performed.

---

## egp

GateD supports the Exterior Gateway Protocol (EGP). EGP is an exterior routing protocol that moves routing information between Autonomous Systems (ASs). Unlike interior protocols, EGP propagates only reachability indications, not true metrics. EGP updates contain metrics, called distances, which range from 0 to 255. GateD only compares EGP distances learned from the same AS. EGP currently has limited usage. By default, EGP is disabled.

Before EGP sends routing information to a remote router, it must establish an adjacency with that router. This occurs by exchanging Hello and I Heard You (I-H-U) messages with that router. (Hello should not be confused with the HELLO protocol, or OSPF HELLO messages.) Computers communicating over EGP are called EGP neighbors, and the exchange of Hello and I-H-U messages is known as acquiring a neighbor.

Once you acquire a neighbor, the system polls it for routing information. The neighbor responds by sending an update containing routing information. If the system receives a poll from its neighbor, it responds with its own update packet. When the system receives an update, it includes routes from the update into its routing database. If the neighbor fails to respond to three consecutive polls, GateD assumes that the neighbor is down and removes the neighbor's routes from its database.

You configure EGP in the `GATED.CONF` file using a GateD protocol statement.

### Format

```
egp yes | no | on | off
[ {
    preference value ;
    defaultmetric metric ;
    packetsize max ;
    traceoptions options ;
    group
        [peeras ASnumber]
        [localas ASnumber]
        [maxup number]
        {
            neighbor host
            [metricout metric]
            [preference value]
            [preference2 value]
            [ttl ttl]
            [nogendefault]
            [importdefault]
            [exportdefault]
            [gateway gateway]
            [lcladdr local-address]
```

```

        [sourcenet network]
        [minhello | p1 time]
        [minpoll | p2 time]
        [traceoptions options] ;
    };
}];

```

## Options and Parameters

**yes | on**

**no | off** (default)

Enables or disables EGP support. Disabled by default.

**preference value ;**

Preference for routes learned from EGP. A preference specified on the `group` or `neighbor` statements or by import policy can override this. The default preference value is 200.

**defaultmetric metric ;**

Metric used when advertising routes over EGP. This choice of values requires you to explicitly specify a metric when exporting routes to EGP neighbors. A metric specified on the `neighbor` or `group` statements or in export policy can override this. The default `metric` is 255.

**packetsize max ;**

Maximum size of a packet that EGP expects to receive from this neighbor. If EGP receives a larger packet, it is incomplete and EGP discards it. EGP notes the length of this packet and increases the expected size to be able to receive a packet of this size. Specifying the parameter prevents the first packet from being dropped. All packet sizes are rounded up to a multiple of the system page size. The default packet size `max` value is 8192.

**traceoptions options ;**

Tracing options for EGP (can be overridden on a group or neighbor basis):

packets	All EGP packets, or packets <code>[detail] send</code> or <code>[detail] rcv</code> ( <code>detail</code> provides a more verbose format to provide more details; if used, <code>detail</code> must come before <code>send</code> or <code>rcv</code> )
---------	---

hello	EGP HELLO/I-HEARD-U packets used to determine neighbor reachability
acquire	EGP ACQUIRE/CEASE packets used to initiate and terminate EGP sessions
update	EGP POLL/UPDATE packets used to request and receive reachability updates

## Group Clause

```

Group [peeras ASnumber] [localas ASnumber] [maxup number]
{
  neighbor host
    [metricout metric]
    [preference value]
    [preference2 value]
    [ttl ttl]
    [nogendefault]
    [importdefault]
    [exportdefault]
    [gateway gateway]
    [lcladdr local-address]
    [sourcenet network]
    [minhello | p1 time]
    [minpoll | p2 time]
    [traceoptions options] ;
};

```

EGP neighbors must be members of a group, which groups all neighbors in one AS. Parameters specified in the `group` clause apply to all the subsidiary neighbors, unless explicitly overridden on a `neighbor` clause. Any number of group clauses can specify any number of neighbor clauses. You can specify any parameters from the `neighbor` subclause on the `group` clause to provide defaults for the whole group (which you can override for individual neighbors).

The `group` clause is the only place to set the following attributes:

### **peeras ASnumber**

AS number expected from peers in the group. Learned dynamically.

### **localas ASnumber**

AS that GateD represents to the group. Usually only used when masquerading as another AS. Use is discouraged. Set globally in `autonomoussystem`.

**maxup number**

Number of neighbors GateD should acquire from this group. GateD attempts to acquire the first maxup neighbors in the order listed. If one of the first neighbors is not available, it acquires one farther down the list. If after startup, GateD does manage to acquire the more desirable neighbor, it drops the less desirable one. By default, GateD acquires all neighbors in the group.

**Group Neighbor Clause**

```
egp yes | no | on | off
[ {
  preference value ;
  defaultmetric metric ;
  packetsize max ;
  traceoptions options ;
  group
    [peeras ASnumber]
    [localas ASnumber]
    [maxup number
    {
      neighbor host
      [metricout metric]
      [preference value]
      [preference2 value]
      [ttl ttl]
      [nogendefault]
      [importdefault]
      [exportdefault]
      [gateway gateway]
      [lcladdr local-address]
      [sourcenet network]
      [p1 time | minhello]
      [p2 time | minpoll]
      [traceoptions options] ;
    }
  ] ;
} ] ;
```

Each neighbor subclause defines one EGP neighbor within a group. The only required part of the subclause is the host argument, the symbolic host name or IP address of the neighbor.

**metricout metric**

Metric used for all routes sent to this neighbor. Overrides the default metric set in the egp statement and any metrics specified by export policy, but only for this specific neighbor or group of neighbors.

**preference value**

Preference used for routes learned from these neighbors. Can differ from the default EGP preference set in the `egp` statement, so that GateD can prefer routes from one neighbor, or group of neighbors, over another. Import policy can explicitly override this.

**preference2 value**

Tie-breaker, in the case of a preference tie. The default `value` is 0.

**ttl ttl**

IPL time-to-live. Provided when attempting to communicate with improperly functioning routers that ignore packets sent with a TTL 1. The default `ttl` for local neighbors is 1; the default for nonlocal neighbors is 255.

**nogendefault**

Does not generate a default route when EGP receives a valid update from its neighbor. The default route is only generated when you enable the `gendefault` option.

**importdefault**

Accepts the default route (0.0.0.0) if included in a received EGP update. For efficiency, some networks have external routers announce a default route to avoid sending large EGP update packets. The default route in the EGP update is ignored.

**exportdefault**

Includes the default route (0.0.0.0) in EGP updates sent to this EGP neighbor. Allows the system to advertise the default route using EGP. Normally a default route is not included in EGP updates.

**gateway gateway**

Router on an attached network used as the next hop router for routes received from this neighbor if a network is not shared with a neighbor. Rarely used.

**lcladdr local-address**

Address used on the local end of the connection with the neighbor. The local address must be on an interface shared with the neighbor, or with the neighbor's gateway when using the `gateway` option. A session only opens when an interface with the appropriate local address (through which the neighbor or gateway address is directly reachable) is operating.

**sourcenet network**

Network queried in the EGP Poll packets. If there is no network shared with the neighbor, specify one of the networks attached to the neighbor. Also use to specify a network shared with the neighbor, other than the one on which the EGP packets are sent. Normally not needed. The default is the network shared with the neighbor's address.

**p1 time** or **minhello**

Minimum acceptable interval between the transmission of EGP HELLO packets. If the neighbor fails to respond to three hello packets, GateD stops trying to acquire the neighbor. Setting a larger interval gives the neighbor a better chance to respond. The `minhello` is an alias for the `p1` value defined in the EGP specification. The default `time` value is 30.

**p2 time** or **minpoll**

Time interval between polls to the neighbor. If three polls are sent without a response, the neighbor is declared "down" and all routes learned from that neighbor are removed from the routing database. A longer polling interval supports a more stable routing database but is not as responsive to routing changes. The `minpoll` is an alias for the `p2` value defined in the EGP specification. The default `time` value is 120.

**traceoptions options**

Tracing options for this EGP neighbor, which are:

packets	All EGP packets, or <code>packets [detail] send</code> or <code>[detail] rcv</code> ( <code>detail</code> provides a more verbose format to provide more details; if used, <code>detail</code> must come before <code>send</code> or <code>rcv</code> )
hello	EGP HELLO/I-HEARD-U packets used to determine neighbor reachability
acquire	EGP ACQUIRE/CEASE packets used to initiate and terminate EGP sessions



update	EGP POLL/UPDATE packets used to request and receive reachability updates
--------	--

---

# bgp

The Border Gateway Protocol (BGP) is an exterior routing protocol used to exchange routing information between multiple transit Autonomous Systems (ASs) as well as between transit and stub ASs. BGP is related to EGP but operates with more capability, greater flexibility, and less bandwidth required. BGP uses path attributes to provide more information about each route. It maintains an AS path, which includes the AS number of each AS the route transits, providing information sufficient to prevent routing loops in an arbitrary topology. You can also use path attributes to distinguish between groups of routes to determine administrative preferences. This allows greater flexibility in determining route preference to achieve a variety of administrative ends.

BGP supports two basic types of sessions between neighbors—internal (sometimes called IBGP) and external. Internal sessions run between routers in the same AS, while external sessions run between routers in different ASs. When sending routes to an external peer, the local AS number is prepended to the AS path. Hence routes received from an external peer are guaranteed to have the AS number of that peer at the start of the path. Routes received from an internal neighbor do not generally have the local AS number prepended to the AS path. Hence, these routes generally have the same AS path the route had when the originating internal neighbor received the route from an external peer. Routes with no AS numbers in the path may be legitimately received from internal neighbors; these indicate that the received route should be considered internal to your own AS.

The BGP implementation supports three versions of the BGP protocol—versions 2, 3 and 4. BGP versions 2 and 3 are similar in capability and function. They only propagate classed network routes, and the AS path is a simple array of AS numbers. BGP version 4 propagates fully general address-and-mask routes, and the AS path has some structure to represent the results of aggregating dissimilar routes.

External BGP sessions may or may not include a single metric, which BGP calls the Multi-Exit Discriminator (MED), in the path attributes. For BGP versions 2 and 3 this metric is a 16-bit unsigned integer; for BGP version 4 it is a 32-bit unsigned integer. In either case, smaller values of the metric are preferred. Currently this metric only breaks ties between routes with equal preference from the same neighbor AS. Internal BGP sessions carry at least one metric in the path attributes, which BGP calls the LocalPref. The size of the metric is identical to the MED. For BGP versions 2 and 3, this metric is better when its value is smaller; for version 4 it is better when it is larger. BGP version 4 sessions optionally carry a second metric on internal sessions, this being an internal version of the MED. The use of these metrics depends on the type of internal protocol processing specified.

BGP collapses routes with similar path attributes into a single update for advertisement. Routes received in a single update are readadvertised in a single update. The churn caused by the loss of a neighbor is minimized, and the initial advertisement sent during peer establishment is maximally compressed. BGP does not read information from the kernel message by message, but fills the input buffer. It processes all complete messages in the buffer before reading again. BGP also does multiple reads to clear all

incoming data queued on the socket. This feature may cause other protocols to be blocked for prolonged intervals by a busy peer connection.

All unreachable messages are collected into a single message and sent prior to reachable routes during a flash update. For these unreachable announcements, the next hop is set to the local address on the connection, no metric is sent, and the path origin is set to incomplete. On external connections the AS path in unreachable announcements is set to the local AS; on internal connections the AS path is set to zero length.

BGP implementation expects external peers to be directly attached to a shared subnet, and expects those peers to advertise next hops that are host addresses on that subnet (although this constraint can be relaxed by configuration for testing). For groups of internal peers, however, there are several alternatives that can be selected by specifying the group type. Type internal groups expect all peers to be directly attached to a shared subnet so that, like external peers, the next hops received in BGP advertisements may be used directly for forwarding. Type routing groups instead determine the immediate next hops for routes, by using the next hop received with a route from a peer as a forwarding address, and using this to look up an immediate next hop in an IGP's routes. Such groups support distant peers, but need to be informed of the IGP whose routes they use to determine immediate next hops. Finally, type IGP groups expect routes from the group peers not to be used for forwarding at all. Instead, they expect that copies of the BGP routes are also received through an IGP, and that the BGP routes are only used to determine the path attributes associated with the IGP routes. Such groups also support distant peers and also need to be informed of the IGP with which they are running.

For internal BGP group types (and for test groups), where possible, a single outgoing message is built for all group peers based on the common policy. A copy of the message is sent to every peer in the group, with possible adjustments to the next hop field as appropriate to each peer. This minimizes the computational load of running large numbers of peers in these types of groups. BGP allows unconfigured peers to connect if an appropriate group was configured with an `allow` clause.

## Format

```
bgp yes | no | on | off
[ {
  preference value ;
  defaultmetric metric ;
  traceoptions options ;
  group type
  external peeras ASnumber
    | internal peeras ASnumber
    | igp peeras ASnumber proto proto
    | routing peeras ASnumber proto proto interface list
    | test peeras ASnumber
}
```

```

allow
{
    network
    network mask mask
    network masklen number
    all
    host host
} ;
peer host
    [metricout metric]
    [localas ASnumber]
    [nogendefault]
    [gateway gateway]
    [preference value]
    [preference2 value]
    [lcladdr local-address]
    [holdtime time]
    [version number]
    [passive]
    [sendbuffer number]
    [recvbuffer number]
    [indelay time]
    [outdelay time]
    [keep [all | none] ]
    [analretentive]
    [noauthcheck]
    [noaggregatorid]
    [keepalivesalways]
    [v3asloopokay]
    [nov4asloop]
    [logupdown]
    [ttl ttl]
    [traceoptions options] ;
};
};

```

## Options and Parameters

**yes | on**

**no | off** (default)

Enables or disables BGP support. Disabled by default.

**preference value ;**

Preference for routes learned from BGP. A preference specified on the `group` or `peer` statements, or by import policy, can override this. The default preference value is 170.

**defaultmetric *metric* ;**

Metric used when advertising routes over BGP. A metric specified on the `group` or `peer` statements, or in export policy, can override this. The default `metric` is 65535.

**traceoptions *options* ;**

Tracing options for BGP. May be overridden on a group or peer basis. The trace `options` are:

<code>packets</code>	All BGP packets, or <code>packets [detail] send or [detail] rcv [detail]</code> provides a more verbose format to provide more details; if used, <code>detail</code> must come before <code>send</code> or <code>rcv</code> ).
<code>open</code>	BGP OPEN packets used to establish a peer relationship
<code>update</code>	BGP UPDATE packets used to pass network reachability information
<code>keepalive</code>	BGP KEEPALIVE packets used to verify peer reachability

## Group Type Clause

**peeras**

For `group type`, specify one of the following `peeras` options:

<code>external peeras ASnumber</code>	In the classic external BGP group, full policy checking is applied to all incoming and outgoing advertisements. The external neighbors must be directly reachable through one of the machine's local interfaces. No metric included in external advertisements and the next hop is computed with respect to the shared interface.
<code>internal peeras ASnumber</code>	Internal group operating where there is no IP-level IGP; for example, an SMDS network or MILNET. All neighbors in this group must be directly reachable over a single interface. All next-hop information is computed with respect to this interface. Import and export policy may be applied to group advertisements. Routes received from external BGP or EGP neighbors are readvertised with the received metric.

<pre>igp peeras ASnumber</pre>	<p>Internal group that runs in association with an interior protocol. The IGP group examines routes the IGP exports, and sends an advertisement only if the path attributes could not be entirely represented in the IGP tag mechanism. Only the AS path, path origin, and transitive optional attributes are sent with routes. No metric is sent, and the next hop is set to the local address the connection uses. Received internal BGP routes are not used or readvertised. Instead, the AS path information is attached to the corresponding IGP route and the latter is used for readvertisement.</p> <p>Since internal IGP peers are sent only a subset of the routes the IGP exports, the export policy used is the IGP's. There is no need to implement the "don't route from peers in the same group" constraint, since the advertised routes are routes that IGP already exports.</p>
<pre>routing peeras ASnumber</pre>	<p>Internal group that uses the routes of an interior protocol to resolve forwarding addresses. A type routing group propagates external routes between routers not directly connected, and computes immediate next hops for these routes by using the BGP next hop that arrived with the route as a forwarding address to be resolved using an internal protocol's routing information.</p> <p>In essence, internal BGP is used to carry AS external routes, while the IGP is expected to only carry AS internal routes, and the latter is used to find immediate next hops for the former. The next hop in BGP routes advertised to the type routing peers are set to local address on BGP connection to those peers, as it is assumed a route to this address is propagated over IGP.</p> <ul style="list-style-type: none"> <li>• <code>proto proto</code> - Interior protocol used to resolve BGP route next hops, and can be the name of any IGP in the configuration.</li> <li>• <code>interface list</code> - Optionally provides a list of interfaces whose routes are carried over the IGP for which third party next hops can be used instead.</li> </ul>
<pre>test peeras ASnumber</pre>	<p>Extension to external BGP that implements a fixed policy using test peers. Fixed policy and special case code make test peers</p>

	<p>relatively inexpensive to maintain. Test peers do not need to be on a directly attached network. If GateD and the peer are on the same (directly attached) subnet, the advertised next hop is computed with respect to that network; otherwise the next hop is the local machine's current next hop.</p> <p>All routing information advertised by and received from a test peer is discarded, and all BGP advertisable routes are sent back to the test peer. Metrics from EGP- and BGP-derived routes are forwarded in the advertisement; otherwise no metric is included.</p>
--	--

## Group Type Allow Clause

Allows peer connections from any addresses in the specified range of network and mask pairs. Configure all parameters for these peers on the group clause. The internal peer structures are created when an incoming open request is received, and destroyed when the connection is broken. (For details on specifying the network/mask pairs, see *Route Filtering*.)

## Group Type Peer Clause

Configures an individual peer. Each peer inherits all parameters specified on a group as defaults. You can override these defaults using parameters explicitly specified in the `peer` subclause. Allows the following parameters:

### **`metricout metric`**

Primary metric on all routes sent to the specified peer(s). Overrides the default metric, a metric specified on the group, and any metric specified by export policy.

### **`localas ASnumber`**

AS that GateD represents to this group of peers. `ASnumber` is set globally in `autonomousssystem`.

### **`nogendefault`**

Does not generate a default route when EGP receives a valid update from its neighbor. The default route is generated only when enabling the `gendefault` option.

**gateway gateway**

If a network is not shared with a peer, specifies a router on an attached network used as the next hop router for routes received from this neighbor. Not needed in most cases.

**preference value**

Preference used for routes learned from these peers. Can differ from the default BGP preference set in the `bgp` statement, so that GateD can prefer routes from one peer, or group of peers, over others. Import policy can explicitly override this.

**preference2 value**

In the case of a preference tie, can break the tie.

**lcladdr local-address**

Address used on the local end of the TCP connection with the peer. For external peers, the local address must be on an interface shared with the peer or with the peer's gateway when using the `gateway` parameter. A session with an external peer only opens when an interface with the appropriate local address (through which the peer or gateway address is directly reachable) is operating. For other types of peers, a peer session is maintained when any interface with the specified local address is operating. In either case, incoming connections are only recognized as matching a configured peer if they are addressed to the configured local address.

**holdtime time**

BGP holdtime value to use when negotiating the connection with this peer, in seconds. According to BGP, if GateD does not receive a keepalive, update, or notification message within the period specified in the Hold Time field of the BGP Open message, the BGP connection is closed. The value must be either 0 (no keepalives are sent) or at least 3.

**version number**

Version of the BGP protocol to use with this peer. If specified, only the specified version is offered during negotiation. Currently supported versions are 2, 3, and 4. By default, the highest supported version is used first, and version negotiation is attempted.



**passive**

Does not attempt active OPENs to this peer. GateD should wait for the peer to issue an open. By default, all explicitly configured peers are active.

**sendbuffer number** and **recvbuffer number**

Controls the amount of send and receive buffering asked of the kernel. The maximum `number` supported is 65535 bytes, although many kernels have a lower limit. Not needed on normally functioning systems. By default, the maximum supported is configured.

**indelay time** and **outdelay time**

Dampens route fluctuations. The `indelay` is the amount of time a route learned from a BGP peer must be stable before it is accepted into the GateD routing database. The `outdelay` is the amount of time a route must be present in the GateD routing database before it is exported to BGP. Default `time` in both cases is 0.

**keep all**

Retains routes learned from a peer even if the routes' AS paths contain one of our exported AS numbers.

**analretentive**

Issues warning messages when receiving questionable BGP updates such as duplicate routes and/or deletions of nonexistent routes. Normally these events are silently ignored.

**noauthcheck**

Communicates with an implementation that uses some form of authentication other than the normal authentication field of all ones.

**noaggregatorid**

GateD should specify the `routerid` in the `aggregator` attribute as zero (instead of its `routerid`) in order to prevent different routers in an AS from creating aggregate routes with different AS paths.

**keepalivesalways**

GateD should always send keepalives, even when an update could have correctly substituted for one. Allows interoperability with routers that do not completely obey the protocol specifications on this point.

**v3asloopokay**

By default, GateD does not advertise routes whose AS path is looped (that have an AS appearing more than once in the path) to version 3 external peers. Setting this flag removes this constraint. Ignored when set on internal groups or peers.

**nov4asloop**

Does not advertise routes with looped AS paths to version 4 external peers. Can be useful to avoid advertising such routes to peer which would incorrectly forward the routes on to version 3 neighbors.

**logupdown**

Logs a message using syslog whenever a BGP peer enters or leaves ESTABLISHED state.

**ttl ttl**

Provided when attempting to communicate with improperly functioning routers that ignore packets sent with a TTL 1. Not all kernels allow the TTL to be specified for TCP connections. The default ttl for local neighbors is 1; the default for nonlocal neighbors is 255.

**traceoptions options ;**

Tracing options for this BGP neighbor include:

packets	All BGP packets, or packets [detail] send or [detail] recv (detail provides a more verbose format to provide more details; if used, detail must come before send or recv)
open	BGP OPEN packets used to establish a peer relationship
update	BGP UPDATE packets used to pass network reachability information
keepalive	BGP KEEPALIVE packets used to verify peer reachability

---

## ospf

Open Shortest Path First (OSPF) routing is a shortest-path-first (SPF) or link-state protocol. OSPF is an interior gateway protocol that distributes routing information between routers in a single Autonomous System (AS). OSPF chooses the least cost path as the best path. Suitable for complex networks with many routers, OSPF provides equal cost multipath routing where packets to a single destination can be sent over more than one interface simultaneously. In a link-state protocol, each router maintains a database describing the entire AS topology, which it builds out of the collected link state advertisements of all routers. Each participating router distributes its local state (that is, the router's usable interfaces and reachable neighbors) throughout the AS by flooding.

Each multiaccess network with at least two attached routers has a designated router and a backup designated router. The designated router floods a link state advertisement for the multiaccess network and has other special responsibilities. The designated router concept reduces the number of adjacencies required on a multiaccess network.

OSPF lets you group networks into areas. Routing information passed between areas is abstracted, which can significantly reduce routing traffic. OSPF uses four different types of routes, listed in order of preference—intra-area, inter-area, type 1 external, and type 2 external. Intra-area paths have destinations within the same area, while inter-area paths have destinations in other OSPF areas. AS External (ASE) routes are routes to destinations external to the AS. Routes imported into OSPF as type 1 routes are supposed to be from IGP's whose external metrics are directly comparable to OSPF metrics.

When making a routing decision, OSPF adds the internal cost of the AS Border router to the external metric. Type 2 ASEs are used for EGP's whose metrics are not comparable to OSPF metrics. In this case, GateD uses only the internal OSPF cost of the AS Border router in the routing decision.

From the topology database, each router constructs a tree of the shortest paths with itself as the root. This shortest-path tree gives the route to each destination in the AS. Externally derived routing information appears on the tree as leaves. The link-state advertisement format distinguishes between information acquired from external sources and from internal routers, so that there is no ambiguity about the source or reliability of routes. Externally derived routing information (for example, routes learned from EGP or BGP) passes transparently through the AS and is separate from OSPF's internally derived data. Each external route can also be tagged by the advertising router, enabling a passing of additional information between routers on the borders of the AS.

OSPF optionally includes type of service (TOS) routing and allows administrators to install multiple routes to a given destination for each type of service (such as for low delay or high throughput.) A router running OSPF uses the destination address and the TOS to choose the best route to the destination.

OSPF intra- and inter-area routes are always imported into the GateD routing database with a preference of 10. It would be a violation of the protocol if an OSPF router did not participate fully in the area's

OSPF, so it is not possible to override this. Although it is possible to give other routes lower preference values explicitly, it is ill-advised to do so.

Hardware multicast capabilities are also used where possible to deliver link-status messages.

OSPF areas are connected by the backbone area, the area with identifier 0.0.0.0. All areas must be logically contiguous and the backbone is no exception. To permit maximum flexibility, OSPF allows the configuration of virtual links to enable the backbone area to appear contiguous when they are actually not.

All routers in an area must agree on that area's parameters. A separate copy of the link-state algorithm is run for each area. Because of this, most configuration parameters are defined on a per area basis. All routers belonging to an area must agree on that area's configuration. Misconfiguration leads to adjacencies not forming between neighbors, and routing information might not flow, or even loop.

**Authentication.** You can authenticate OSPF protocol exchanges. Authentication guarantees that routing information is imported only from trusted routers, to protect the Internet and its users. There are two authentication schemes available. The first uses a simple authentication key of up to eight characters and is standardized. The second is still experimental and uses the MD5 algorithm and an authentication key of up to 16 characters.

The simple password provides very little protection, because in many cases it is possible to easily capture packets from the network and learn the authentication key. The experimental MD5 algorithm provides much more protection, as it does not include the authentication key in the packet.

The OSPF specification currently specifies that you configure the authentication type per area with the ability to configure separate passwords per interface. This was extended to allow configuration of different authentication types and keys per interface. Also, you can specify both a primary and a secondary authentication type and key on each interface. Outgoing packets use the primary authentication type, but incoming packets may match either the primary or secondary authentication type and key.

You configure OSPF in the `MULTINET:GATED.CONF` file using a GateD protocol statement.

## Format

```
ospf yes | no | on | off
[ {
  defaults
  {
    preference value ;
    cost cost ;
    tag [as] tag ;
    type 1 | type 2 ;
  } ;
  exportlimit routes ;
```

```

exportinterval time ;
traceoptions options;
monitorauthkey key ;
monitorauth none | [simple | md5] authkey ;
backbone | area area
{
  authtype 0 | authtype 1 | none | simple ;
  stub [cost cost] ;
  networks
  {
    network [restrict] ;
    network mask mask [restrict] ;
    network masklen number [restrict] ;
    host host [restrict] ;
  } ;
  stubhosts
  { host cost cost ; } ;
  interface list [cost cost]
  { interface-parameters } ;
  interface list nonbroadcast [cost cost]
  {
    pollinterval time ;
    routers
    {
      gateway [eligible] ;
    };
    interface-parameters
  } ;
  /* Backbone only: */
  virtuallink neighborid router-id transitarea area
  { interface-parameters } ;
} ;
}] ;

```

## Options and Parameters

**yes | on**

**no | off**

Enables or disables OSPF support.

### defaults

Defaults used when importing OSPF ASE routes into the GateD routing table, and exporting routes from the GateD routing table into OSPF ASEs, including:

<code>preference value;</code>	How OSPF routes compete with routes from other protocols in the GateD routing table. The default preference value is 150.
--------------------------------	---

<code>cost cost ;</code>	Used when exporting a non-OSPF route from the GateD routing table into OSPF as an ASE. Export policy can explicitly override this. The default <code>cost</code> is 1.
<code>tag [as] tag ;</code>	OSPF ASE routes have a 32-bit tag field that the OSPF protocol does not use, but export policy can use it to filter routes. When OSPF interacts with an EGP, you can use the tag field to propagate AS path information. In this case you would specify the <code>as</code> keyword and the tag is limited to 12 bits of information. The default <code>tag</code> value is 0.
<code>type 1 or 2 ;</code>	Export policy can explicitly change and override the default here. The default is <code>type 1</code> .

**`exportlimit routes ;`**

How many ASEs are generated and flooded in each batch. The default `export limits routes` value is 100.

**`exportinterval time ;`**

How often a batch of ASE link state advertisements are generated and flooded into OSPF. The default `export interval time` value is 1 (once per second).

**`traceoptions options ;`**

In addition to the following OSPF specific trace flags, OSPF supports the state which traces interface and neighbor state machine transitions:

<code>lsabuild</code>	Link State Advertisement creation
<code>spf</code>	Shortest Path First (SPF) calculations
<code>lsatransmit</code>	Link State Advertisement (LSA) transmission
<code>lsareceive</code>	LSA reception
<code>state</code>	State transitions

Packet tracing options (which you can modify with `detail`, `send`, and `recv`):

hello	OSPF HELLO packets used to determine neighbor reachability
dd	OSPF Database Description packets used in synchronizing OSPF databases
request	OSPF Link State Request packets used in synchronizing OSPF databases
lsu	OSPF Link State Update packets used in synchronizing OSPF databases
ack	OSPF Link State Ack packets used in synchronizing OSPF databases

<pre> authtype 0 or 1 or none or simple </pre>	<p>OSPF specifies an authentication scheme per area. Each interface in the area must use this same authentication scheme, although it can use a different authentication key. 0 is the same as none; 1 is the same as simple.</p>
<pre> stub [cost cost] </pre>	<p>A stub area is one in which there are no ASE routes. Use cost to inject a default route into the area with the specified cost.</p>
<pre> networks { network [restrict] ;   network mask mask [restrict] ;   network masklen number [restrict];   host host [restrict] ; } ; </pre>	<p>The <code>networks</code> list describes the scope of an area. Intra-area LSAs that fall within the specified ranges are not advertised into other areas as inter-area routes. Instead, the specified ranges are advertised as summary network LSAs.</p> <p>If you specify <code>restrict</code>, the summary network LSAs are not advertised. Intra-area LSAs that do not fall into any range are also advertised as summary network LSAs. This option is very useful on well-designed networks in reducing the amount of routing information propagated between areas. The entries in this list are either networks, or a subnetwork/mask pair.</p>



<pre>stubhosts { host cost cost ; }</pre>	<p>The <code>stubhosts</code> list specifies directly attached hosts that should be advertised as reachable from this router, and the costs with which they should be advertised. Specify point-to-point interfaces here on which it is not desirable to run OSPF.</p> <p>It is also useful to assign an additional address to the loopback interface (one not on the 127 network) and advertise it as a stub host. If this address is the same one used as the router ID, it enables routing to OSPF routers by router ID, instead of by interface address. This is more reliable than routing to one of the router's interface addresses, which may not always be reachable.</p>
<pre>interface list cost cost {interface-parameters}</pre>	<p>Use this form of the <code>interface</code> clause (with the optional <code>cost</code> value, and immediately followed by the <code>interface-parameters</code>) to configure a broadcast (which requires IP multicast support) or a point-to-point interface. (See the <code>interfaces</code> statement for a description of <code>list</code>.) Each interface has a cost. The costs of all the interfaces a packet must cross to reach a destination are summed to get the cost to that destination. The <code>cost</code> can be any non-zero value (the default is 1).</p>

**monitorauthkey key ;**

**monitorauth none | [simple | md5] authkey ;**

You can query the OSPF state using the `ospf_monitor` utility, which sends nonstandard OSPF packets that generate a text response from OSPF. If you configure an authentication key, the incoming requests must match the specified authentication key. These packets cannot change OSPF state, but the act of querying OSPF can expend system resources. Not authenticated by default.

## Backbone/Area Clause Options and Parameters

**backbone** or **area area**

Configures each OSPF router into at least one OSPF area. If you configure more than one area, at least one must be the backbone. Configure the backbone using the `backbone` keyword only; you cannot specify it as `area 0`. The backbone interface can be a `virtuallink`.

Further parameters include:

The following are the interface-parameters. You can specify them on any class of interface:

```
enable | disable ;  
retransmitinterval time ;  
transitdelay time ;  
priority value ;  
hellointerval time ;  
routerdeadinterval time ;  
authkey key ;
```

retransmitinterval <i>time</i>	Number of seconds between link state advertisement retransmissions for adjacencies belonging to this interface.
transitdelay <i>time</i>	Estimated number of seconds required to transmit a link state update over this interface. Takes into account transmission and propagation delays and must be greater than 0.
priority <i>value</i>	Number between 0 and 255 specifying the priority for becoming the designated router on this interface. When two routers attached to a network both attempt to become designated router, the one with the highest priority prevails. A router whose router priority is 0 is ineligible to become designated router.
hellointerval <i>time</i>	Length of time, in seconds, between Hello packets that the router sends on the interface.
routerdeadinterval <i>time</i>	Number of seconds not hearing a router's Hello packets before the router's neighbors will declare it down.
authkey <i>key</i>	Used by OSPF authentication to generate and verify the authentication field in the OSPF header. You can configure the authentication key on a per-interface basis. Specify it using one to eight decimal digits separated by periods, a one to eight byte hexadecimal string preceded by 0x, or a one to eight character string in double quotes.

The form of the `interface` clause with the `nobroadcast` option is for point-to-point interfaces only. By default, OSPF packets to neighbors on point-to-point interfaces are sent using the IP multicast mechanism. GateD detects this condition and falls back to using sending unicast OSPF packets to this point-to-point neighbor.

If you do not want IP multicasting, because the remote neighbor does not support it, specify `nobroadcast` to force the use of unicast OSPF packets. You can also use this option to eliminate warnings when GateD detects the bug mentioned previously. (See the previous page for the `interface-parameters`.)

Use this form of the `interface` clause to specify a nonbroadcast interface on a nonbroadcast multiaccess (NBMA) media. Since an OSPF broadcast media must support IP multicasting, you must configure a broadcast-capable media, such as Ethernet, that does not support IP multicasting as a nonbroadcast interface. A nonbroadcast interface supports any of the standard interface clauses listed previously, plus the following two that are specific to nonbroadcast interfaces:

<code>pollinterval time</code>	Before adjacency is established with a neighbor, OSPF packets are sent periodically at the specified poll interval.
<code>routers gateway</code>	By definition, it is not possible to send broadcast packets to discover OSPF neighbors on a nonbroadcast, so you must configure all neighbors. The list includes one or more neighbors and an indication of their eligibility to become a designated router.

**`virtuallink neighborid routerid transitarea area`  
`{ interface-parameters } ;`**

For backbone only: Virtual links are used to establish or increase connectivity of the backbone area. The `neighborid` is the router-ID of the other end of the virtual link. The transit area specified must also be configured on this system. You can specify all standard interface parameters defined by the interface clause previously described on a virtual link. (See the previous page for the `interface-parameters`.)

---

# static

The `static` statements define the static routes GateD uses. A single `static` statement can specify any number of routes. These statements must occur after `protocol` statements and before `control` statements in `GATED.CONF`. Specify any number of `static` statements, each containing any number of static route definitions. You can override these routes with ones with better preference values.

## Format

```
static
{
  host host gateway list
    | network [mask mask | masklen number] gateway list
    | default gateway list
      [interface list]
      [preference value]
      [retain]
      [reject]
      [blackhole]
      [noinstall]
    ;
  network [mask mask | masklen number]
    interface interface
      [preference value]
      [retain]
      [noinstall]
    ;
} ;
```

## Options and Parameters

### **host...gateway *list*** or **default gateway *list***

Most general form of the static statement. Defines a static route through one or more gateways. Static routes are installed when one or more of the gateways listed are available on directly attached interfaces. If more than one eligible gateway is available, they are limited by the number of multipath destinations supported.

The second form of the `network mask...` clause farther down in the statement is for primitive support of multiple network addresses on one interface.

### **interface *list***

Gateways are valid only when they are on one of these interfaces.

**preference value**

Preference of this static route. Controls how this route competes with routes from other protocols. The default value is 60.

**retain**

Prevent specific static routes from being removed. Normally GateD removes all routes except interface routes from the kernel forwarding table during a graceful shutdown. Useful for ensuring that some routing is available when GateD is down.

**noinstall**

Do not install the route in the kernel forwarding table when active, but make it still exportable to other protocols. Normally the route with the lowest preference is installed there and is the route exported to other protocols.

---

# import

The control statements are:

```
import
export
aggregate
generate
```

## Format

```
import [restrict | preference value]
```

The `import` statements control importing routes from routing protocols, and installing the routes in GateD's routing database. The format of an `import` statement varies depending on the source protocol. In all cases, you can specify one of two keywords to control how routes compete with other protocols:

<code>restrict</code>	Restrict the routes from the routing table. In some cases this means that the routes are not installed in the routing table. In others, it means that they are installed with a negative preference; this prevents them from becoming active so that they will not be installed in the forwarding table or exported to other protocols.
<code>preference value</code>	Preference value used when comparing this route to other routes from other protocols. The route with the lowest preference available at any given route becomes the active route, is installed in the forwarding table, and can be exported to other protocols. The individual protocols configure the default preferences.

## Importing Routes from BGP and EGP

You can control EGP importation by AS. Note that EGP and BGP versions 2 and 3 only support propagating natural networks, so the host and default route filters are meaningless. BGP version 4 supports propagating any destination along with a contiguous network mask.

EGP and BGP both store any routes rejected implicitly by their not being mentioned in a route filter, or explicitly if `restrict` appears in the routing table with a negative preference. A negative preference prevents a route from becoming active, which prevents it from being installed in the forwarding table or exported to other protocols. This removes the need to break and reestablish a session on reconfiguring if changing the importation policy.

The syntax of the `import` statement for importing routes from BGP or EGP is any of the following:

```
import proto bgp | egp autonomoussystem ASnumber restrict ;
import proto bgp | egp autonomoussystem ASnumber
  [preference value] {
  route-filter [restrict | preference value] ; } ;
import proto bgp aspath ASpathregex
  origin any | [igp] [egp] [incomplete] restrict ;
import proto bgp aspath ASpathregex
  origin any | [igp] [egp] [incomplete]
    [preference value] {
    routefilter [restrict | preference value] ; } ;
```

The third and fourth variation of the `import` statements is for BGP only and supports controlling propagation by using AS path regular expressions. An AS path is a list of ASs that routing information passes through to get to a router, and an indicator of the origin of the AS path. Use this information to set the preference of one path to a destination network over another. You do this by listing patterns applied to AS paths when importing and exporting routes. Each AS that a route passes through prepends its AS number to the beginning of the AS path.

## Aspath Clause

The following `aspath` clause in the `import` statement indicates that an AS matching the `ASpathregex` with the specified origin is matched. The parameters follow:

```
aspath ASpathregex origin any | [igp] [egp] [incomplete]
```

## Aspath Clause Regular Expression

### ASpathregex

Regular expression, with the alphabet as the set of AS numbers, consisting of one or more AS path expressions, which are terms and operators. An AS path term (`ASpathterm`) consists of the following:

<code>ASnumber</code>	Any valid AS system number, from 1 through 65534.
<code>.</code>	Matches any AS number.
<code>(ASpathregex)</code>	Parentheses group sub-expressions. An operator such as asterisk (*) or question mark (?) works on a single element or on a regular expression enclosed in parentheses.

## ASpath Clause Operators

AS path operators consists of the following:

ASpathterm { <i>m</i> }	Exactly <i>m</i> repetitions, where <i>m</i> is a positive integer.
ASpathterm { <i>m</i> , }	<i>m</i> or more repetitions, where <i>m</i> is a positive integer.
ASpathterm { <i>m</i> , <i>n</i> }	At least <i>m</i> and at most <i>n</i> repetitions, where <i>m</i> and <i>n</i> are both nonnegative integers and $m \leq n$ .
ASpathterm *	Zero or more repetitions (shorthand for {0, }).
ASpathterm +	One or more repetitions (shorthand for {1, }).
ASpathterm ?	Zero or one repetition (shorthand for {0, 1}).
ASpathterm   ASpathterm	Matches either term.

## Remaining Import Statement Options

```
origin any | [igp] [egp] [incomplete]
```

Details the completeness of AS path information. An origin of `igp` indicates that the route was learned from an interior routing protocol and is most likely complete. An origin of `egp` indicates that the route was learned from an exterior routing protocol that does not support AS paths (EGP for example), and that the path is most likely not complete. When the path information is definitely not complete, use `incomplete`.

## Importing Routes from RIP, HELLO, and Redirects

You can control importing RIP, HELLO, and Redirect routes by any protocol, source interface, or source gateway. If using more than one, they are processed from most general (protocol) to most specific (gateway). RIP and HELLO do not support preferences to choose between routes of the same protocol; they use metrics instead. They also do not save rejected routes since they have short update intervals.

The syntax of the `import` statement for importing routes from RIP, HELLO, or redirects is either of the following:



```
import proto rip | hello | redirect
    [interface list | gateway list]
    restrict ;

import proto rip | hello | redirect
    [interface list | gateway list]
    [preference value]
    { routefilter [restrict | preference value] ; } ;
```

## Importing Routes from OSPF

You can only control importing AS External (ASE) routes. OSPF intra- and inter-area routes are always imported into the GateD routing table with a `preference` of 10. If using an `ospftag`, the import clause only applies to routes with the specified tag.

You can only restrict importing OSPF ASE routes if functioning as an AS border router. Do this by specifying an `export ospfase` clause. Specifying an empty export clause can restrict importing ASEs, when no ASEs are exported.

Like the other interior protocols, you cannot use `preference` to choose between OSPF ASE routes; OSPF costs accomplish this. Routes rejected by policy go into the table with a negative preference.

The syntax of the `import` statement for importing routes from OSPF is either of the following:

```
import proto ospfase [tag ospftag] restrict ;

import proto ospfase [tag ospftag]
    [preference value]
    { routefilter [restrict | preference value] ; } ;
```

---

# export

## Format

```
export [restrict | metric metric]
```

The `export` statement controls which routes GateD advertises to other systems. Like `import`, the `export` syntax varies slightly for each protocol. Both syntaxes are similar and the meanings of many of the parameters are the same. The main difference is that while source information controls importing routes, both destination and source information control exporting routes.

The outer portion of a given `export` statement specifies the destination of the routing information you control. The middle portion restricts the sources. The innermost portion is a route filter used to select individual routes.

One thing that applies in all cases is the specification of a metric. All protocols define a default metric for routes exported. In most cases, this can be overridden at several levels of the export statement. The most specific specification of a metric is the one applied to the route exported. The values you can specify for a metric depend on the destination protocol the `export` statement references:

<code>restrict</code>	Do not export anything. If specified on the destination portion of the <code>export</code> statement, it means not to export anything to this destination. If specified on the source portion, it means not to export anything from this source. If specified as part of a route filter, it means not to export the routes matching that filter.
<code>metric <i>metric</i></code>	Metric used when exporting to the specified destination.

## Exporting to EGP and BGP

The AS controls exporting to EGP and BGP, the same policy applied to all routers in the AS. EGP metrics range from 0 through 255, with 0 the most attractive. BGP metrics are 16-bit unsigned quantities (that range from 0 through 65535, inclusive with 0 the most attractive). While BGP version 4 actually supports 32-bit unsigned quantities, GateD does not yet support this.

If you do not specify an export policy, only routes to attached interfaces are exported. If you specify any policy, the defaults are overridden; you should explicitly specify everything you want exported. (Note that EGP and BGP versions 2 and 3 only support the propagation of natural networks, so the host and default route filters are meaningless. BGP version 4 supports the propagation of any destination along with a contiguous network mask.)

The syntax of the `export` statement for exporting routes to EGP or BGP is either of the following:

```
export proto  bgp | egp  as ASnumber  restrict ;
export proto  bgp | egp  as ASnumber  [metric metric]
    {  exportlist  ;  } ;
```

## Exporting to RIP and HELLO

Any protocol, interface, or gateway can control exporting to RIP and HELLO. If you specify more than one, they are processed from most general (protocol) to most specific (gateway). It is not possible to set metrics for exporting RIP routes into RIP, or exporting HELLO routes into HELLO. Attempts to do this are silently ignored.

If you do not specify an export policy, RIP and interface routes are exported into RIP and HELLO, and interface routes are exported into HELLO. If you specify any policy, the defaults are overridden; it is necessary to explicitly specify everything that should be exported.

RIP version 1 and HELLO assume that all subnets of the shared network have the same subnet mask, so they are only able to propagate subnets of that network. RIP version 2 is capable of propagating all routes, when not sending version 1 compatible updates.

To announce routes that specify a next hop of the loopback interface (static and internally generated default routes) over RIP or HELLO, specify the metric at some level in the export clause. Just setting a default metric is not sufficient. This is a safeguard to verify that the announcement is intended.

The syntax of the `export` statement for exporting routes to RIP or HELLO is either of the following:

```
export proto  rip | hello
    [interface list  |  gateway list] restrict ;

export proto  rip | hello
    [interface list  |  gateway list]  [metric metric]
    {  exportlist  ;  } ;
```

## Exporting to OSPF

It is not possible to create OSPF intra- or inter-area routes by exporting routes from the GateD routing table into OSPF. It is only possible to export from the GateD routing table into OSPF ASE routes. It is also not possible to control the propagation of OSPF routes within the OSPF protocol.

There are two types of OSPF ASE routes, type 1 and type 2 (see the OSPF protocol configuration for details on the two types). Specify the default type using the `defaults` subclause of the `ospf` clause. You can override this with the `export` statement.

OSPF ASE routes also have the provision to carry a tag. This is an arbitrary 32-bit number you can use on OSPF routers to filter routing information. (See the OSPF protocol configuration for details on OSPF tags.) You can override the default tag specified by the `ospf defaults` clause with a tag specified on the `export` statement.

The syntax of the `export` statement for exporting routes to OSPF is either of the following:

```
export proto ospfase [type 1 | 2] [tag ospf-tag] restrict ;

export proto ospfase [type 1 | 2] [tag ospf-tag]
    [metric metric]
    { exportlist ; } ;
```

## Exporting BGP and EGP Routes

You can specify BGP and EGP routes by source AS. You can export all routes by AS path. The syntax of the `proto` statement for exporting BGP or EGP routes is either of the following:

```
proto bgp | egp autonomoussystem ASnumber restrict ;
proto bgp | egp autonomoussystem ASnumber [metric metric]
    { routefilter [restrict | metric metric] ; } ;
```

## Exporting RIP and HELLO Routes

You can export RIP and HELLO routes by protocol, source interface, or source gateway. The syntax of the `proto` statement for exporting RIP or HELLO routes is either of the following:

```
proto rip | hello
    [interface list | gateway list] restrict ;
proto rip | hello
    [interface list | gateway list] [metric metric]
    { routefilter [restrict | metric metric] ; } ;
```

## Exporting OSPF Routes

You can export both OSPF and OSPF ASE routes into other protocols. The syntax of the `proto` statement for exporting OSPF routes is either of the following:

```
proto ospfase | ospfase restrict ;
proto ospfase | ospfase [metric metric]
    { routefilter [restrict | metric metric] ; } ;
```

## Exporting Routes from Nonrouting Protocols with Interface

If you want GateD to export direct or static routes, or routes learned from the kernel, use the protocol statement or interface statement along with the interface of the next hop in the GateD configuration file. The syntax of the `proto` statement for exporting routes from nonrouting protocols with an interface is either of the following:

```
proto direct | static | kernel
    [interface list] restrict ;

proto direct | static | kernel
    [interface list] [metric metric]
    { routefilter [restrict | metric metric] ; } ;
```

The `proto` statement parameters include:

direct	Routes to directly attached interfaces.
static	Static routes specified in a static clause.
kernel	On systems with the routing socket, routes learned from the routing socket are installed in the GateD routing table with a protocol of <code>kernel</code> . You can export these routes by referencing this protocol. This is useful when it is desirable to have a script install routes with the <code>route</code> command and propagate them to other routing protocols.

## Exporting Routes from Non-Routing Protocols by Protocol

If you want GateD to export default or aggregate routes, use the protocol statement in the GateD configuration file. The syntax of the `proto` statement for exporting routes from nonrouting protocols by protocol is either of the following:

```
proto default | aggregate restrict ;
proto default | aggregate
    [metric metric]
    { routefilter [restrict | metric metric] ; } ;
```

The `proto` statement parameters include:

default	Routes created by the <code>gendefault</code> option. Use route generation instead.
aggregate	Routes synthesized from other routes when using the <code>aggregate</code> and <code>generate</code> statements.

## Exporting by AS Path

When configuring BGP, all routes get an AS path when added to the routing table. For all interior routes, this AS path specifies IGP as the origin and no ASEs in the AS path (the current AS is added when the route is exported). For EGP routes, this AS path specifies EGP as the origin and the source AS as the AS path. For BGP routes, the AS path is stored as learned from BGP. (The AS path regular expression syntax appears in the *Importing Routes from BGP and EGP* subsection.)

The syntax of the `proto` statement for exporting by AS path is either of the following:

```
proto proto | all aspath ASpathregex
  origin any | [igp] [egp] [incomplete] restrict ;
proto proto | all aspath ASpathregex
  origin any | [igp] [egp] [incomplete] [metric metric]
  { routefilter [restrict | metric metric] ; } ;
```

## Exporting by Route Tag

Both OSPF and RIP version 2 currently support tags. All other protocols always have a tag of zero. You can select the source of exported routes based on this tag. This is useful when classifying routes by tag when exporting them into a given routing protocol. The syntax of the `proto` statement for exporting by route tag is either of the following:

```
proto proto | all all tag tag restrict ;
proto proto | all all tag tag
  [metric metric]
  { routefilter [restrict | metric metric] ; } ;
```

---

# aggregate

Use route aggregation to generate a more general route from a specific one. Use it, for example, at an AS border to generate a route to a network to be advertised through EGP, given the presence of one or more subnets of that network learned through RIP. Regional and national networks also use route aggregation to reduce routing information. By carefully allocating network addresses to clients, regional networks can just announce one route to regional networks instead of hundreds. No aggregation occurs unless explicitly requested in an `aggregate` statement.

Aggregate routes are not actually used for packet forwarding by the originator of the aggregate route, only by the receiver (if it wishes). A router, receiving a packet that does not match one of the component routes that led to the generation of an aggregate route, is supposed to respond with an ICMP `network unreachable` message. This prevents packets for unknown component routes from following a default route into another network where they would be continuously forwarded back to the border router, until their TTL expires. Sending an unreachable message for a missing piece of an aggregate is only possible on systems that support reject routes, which MultiNet does not.

## Format

```
aggregate default | network [mask mask | masklen number]
    [preference value] [brief]
    { proto [all | direct | static | kernel | aggregate | proto]
      [as AS | tag tag | aspath ASpathregex] restrict ;
    proto [all | direct | static | kernel | aggregate | proto]
      [as AS | tag tag | aspath ASpathregex] [preference value]
      { routefilter [restrict | preference value] ; } ;
  } ;
```

## Options and Parameters

### **preference value**

The default preference value is 130.

### **brief**

Truncate the AS path to the longest common AS path. The default is to build an AS path consisting of SETs and SEQUENCES of all contributing AS paths.

**proto *proto***

In addition to the special protocols listed, you can select the contributing protocol from among those currently configured in GateD.

**as *AS***

Restrict selection of routes to those learned from the specified AS.

**tag *tag***

Restrict selection of routes to those with the specified tag.

**aspath *ASpathregex***

Restrict selection of routes to those that match the specified AS path.

**restrict**

Restrict certain routes from contributing to the specified aggregate.

A route can only contribute to an aggregate route that is more general than itself; it must match the aggregate under its mask. Any given route can only contribute to one aggregate route, which will be the most specific configured, but an aggregate route can contribute to a more general aggregate.

---



# generate

A slight variation on aggregation is generating a route based on certain conditions. This is sometimes known as the "route of last resort." This route inherits the next hops and AS path from the contributor specified with the lowest (most favorable) preference. The most common usage is to generate a default based on the presence of a route from a peer on a neighboring backbone.

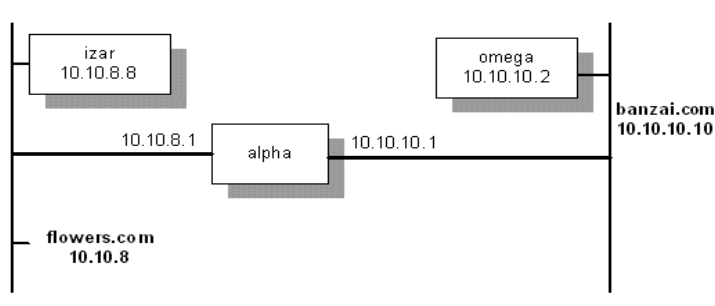
## Format

```
generate default | network [mask mask | masklen number]
    [preference value] [brief]
    { [as AS | tag tag | aspath ASpathregexp]
      restrict ;
    proto [all | direct | static | kernel | aggregate | proto]
    [as AS | tag tag | aspath ASpathregexp]
    [preference value]
    { routefilter [restrict | preference value] ; } ;
} ;
```

---

# Sample GateD Configurations

The below diagram shows two networks connected within an AS using RIP. The following configuration files show the RIP statements on each end host and gateway alpha, which has IP forwarding enabled. All systems are running GateD.



## Configuration File for izar

```
# turn on RIP and listen for updates.
#
rip on;
```

## GateD Configuration File for alpha

```
# turn on RIP.
#
rip yes;
#
# use RIP to pass routing information to the banzai network.
#
export proto rip interface 10.10.10.1
{
    # we know about the flowers network, so announce it.
    #
    proto direct {
        10.10.8.0 mask 255.255.255.0;
    };
    # use RIP to announce all routes learned from flowers.
    #
    proto rip interface 10.10.8.0 {
        all;
    };
};
```

## GateD Configuration File for omega

```
# turn on RIP and listen for updates.
#
rip on;
```

Below is a sample RIP statement where the gateway announces a default route to the backbone, and announces all of the individual subnet routes to the outside world.

```
# enable RIP:
#
rip yes;

# using RIP, announce all local subnets via interface 192.168.12.3:
#
export proto rip interface 192.168.12.3 metric 3
{
    proto rip interface 192.168.1.5
    {
        all;
    };
};

#
# Using RIP, announce default via interface 192.168.1.5:
#
export proto rip interface 192.168.3.1
{
    proto rip interface 192.168.1.5
    {
        default;
    };
};
```

Below is a configuration for AS 283 that enables RIP and OSPF, which you can use to test both.

```
# this interface is passive:
#
interfaces {
    interface SVA-0 passive;
};

#
# this Autonomous System number is 283:
#
autonomous system 283;

#
# turn on RIP:
# packets are to be broadcast.
# metric for routes learned via other protocols is 5.
# multicast RIP V2 packets on SVA-0.
#
rip yes {
    broadcast;
```

```

        defaultmetric 5;
        interface SVA-0 version 2 multicast;
    };

#
# turn on OSPF:
#   Trace Link State Advertisement creation and
#   Shortest Path First calculations
#   use authentication key "ZZZZZZZZ" when handling OSPF queries.
#   this system is on the backbone.
#   use simple password authentication for this area.
#   make this system very unlikely to be a designated router.
#   set the OSPF header authentication key to "YYYYYYYY" for
#   packets going out on SVA-0.
#
ospf yes {
    traceoptions lsabuild spf;
    monauthkey "ZZZZZZZZ";
    backbone {
        authtype simple;
        interface all {
            priority 2;
        };
        interface SVA-0 {
            authkey "YYYYYYYY";
        };
    };
};

```

Below is a configuration for a static route.

```

#
# in this example our host's address is 192.168.1.42
#
static {
    192.168.2.0 masklen 24 interface 192.168.1.42 retain;
    default gateway 192.168.1.1 ;
};

```

---

# RTSOLD

RTSOLD is the IPv6 router solicitation daemon. It will be automatically run at startup when IPv6 has been enabled for an interface. RTSOLD sends ICMPv6 Router Solicitation messages on the specified interface. The ICMPv6 code will use the information in Router Advertisement messages that it receives to configure interface prefix and default route. IPv6 Routers send out advertisement messages periodically and when requested; usage of RTSOLD reduces the amount of time until a router advertisement message is processed by the system and IPv6 routes are defined. RTSOLD will send out solicitation messages until a Router Advertisement message has been received or three attempts have been made.

## Format

```
rtsold [-adDF] [interface]
```

## Options:

- a - auto interface probe. RTSOLD determines what interfaces are available and sends out solicitation messages on the interfaces that it finds.
- d - debug level 1
- D - debug level 2, includes timer information
- F - Force setting of kernel variables. IPV6CTL\_ACCEPT\_RTADV is set to 1 and IPV6CTL\_FORWARDING is set to 0 (zero). This allows Router Advertisement messages to be processed and disables forwarding (routing) on this system.

MULTINET:START\_MULTINET.COM will start RTSOLD with -a -F when an IPv6 interface has been configured.

# 11. Network Time Protocol (NTP)

NTP is the application set for network time protocol functions.

## Converting from earlier versions of NTP

NTP is backwards compatible with prior NTP protocol versions back to version 2, and version 1 in client/server mode. This allows other nodes in the network to be upgraded at different times with minimal disruption.

MultiNet's NTP implementation is based on Network Time Protocol Version 4.2.

## Overview of NTP

The standard timescale used by most nations of the world is Coordinated Universal Time (UTC), which is based on the Earth's rotation about its axis, and the Gregorian Calendar, which is based on the Earth's rotation about the Sun. UTC time is disseminated by various means, including radio and satellite navigation systems, telephone modems and portable clocks. For reasons of cost and convenience, it is not possible to equip every computer with a method of receiving these time signals directly. However, it is possible to equip some number of computers acting as primary time servers to synchronize a much larger number of secondary servers and clients connected by a common network. In order to do this, a distributed network clock synchronization protocol is required which can transmit an accurate reading to one or more clients and adjust each client clock as required. This is what the Network Time Protocol is for.

The synchronization protocol determines the time offset of the server clock relative to the client clock. On request, the server sends a message including the time the request arrived and the time the response was returned. The client included the time it sent the request in the request message, and records the time the response arrived back from the server as well. With these four values, or *timestamps*, the client can determine the server-client propagation delay (by assuming that this is half the round-trip time) and

subtract this from the time difference between client and server time settings to determine its clock offset relative to the server. In general, this is a useful approximation; however, in the Internet of today, network paths and the associated delays can differ significantly due to the individual service providers, and this can contribute to error in determining offset. A stable, symmetrical (in terms of propagation delay) and reliable connection to the time server is important in minimizing this type of error.

NTP attempts to compensate for the problem of network instability by allowing the use of several servers as time sources and determining which of them is most reliable through statistical means that compare them to each other. All sources are assumed to have correct times, but those that differ markedly from the group are eventually ignored as having unreliable connections, or being otherwise poor sources of correct time information. This tends to limit malicious activities as well, where a server that reports false times is inserted in a network, as well as bad time servers that result from hardware failure that can have much the same effect.

Clock errors can be due to other causes than variations in network delay. Other causes include latencies in computer hardware and software (jitter), as well as clock oscillator instability (wander). Despite these sources of error, NTP can, over many updates, discipline a clock to stay remarkably close to the actual time, even when a time server is not available for some period.

## Programs and Files

There are several programs and files that make up NTP in MultiNet. These are described in more detail later in this chapter.

### Program Files

The following programs make up the NTP implementation in MultiNet:

NTPD	The NTP server process used to maintain the system clock and to pass time information to lower stratum clients and servers. While this program runs as a server process, it also functions as a client in requesting time data from other servers on the network.
NTPDATE	An interactive “one shot” time setting utility. It is useful for setting the initial time on a system, perhaps at boot time, to minimize the correction necessary with NTPD. If NTPD

	is not desired for some reason, NTPDATE used in a recurring batch job can be used to maintain a system's clock fairly accurately.
NTPDC	NTPDC is used to query the NTPD server about its current state and to request changes in that state. Extensive state and statistics information is available through the NTPDC interface. In addition, nearly all the configuration options which can be specified at startup using NTPD's configuration file may also be specified at run time using NTPDC.
NTPQ	The NTPQ utility program is used to query NTP servers about current state and to request changes in that state. Requests to read and write arbitrary variables can be assembled, with raw and pretty-printed output options being available. NTPQ can also obtain and print a list of peers in a common format by sending multiple queries to the server. NTPQ and NTPDC perform similar functions, but use different protocols to communicate with NTPD.
NTPTRACE	NTPTRACE determines where a given NTPD server gets its time, and follows the chain of NTP servers back to their master time source.

## Configuration Files

NTP uses the following configuration files:

NTP.CONF	MULTINET:NTP.CONF is used to specify servers from which time information is requested as well as many other aspects of NTPD behavior. See the description of NTPD for a list of options and their definitions. NTP.CONF is read only at NTPD startup, so if you make changes you will need to restart NTPD.
NTP.KEYS	MULTINET:NTP.KEYS is used to define security information used in authorization operations. (The keys used by the NTPQ and NTPDC programs are checked against passwords requested by the programs and entered by hand.)
TIMEZONES.DAT	A default set of timezone rules is compiled into NTP. You can use the MULTINET SET/TIMEZONE command in conjunction with this file to add other timezone rules. See the <i>Timezone</i> section for more information about timezones and the use of this file.



## Other Files

NTP uses the following files:

NTPD.LOG	The NTPD server outputs progress and error information to the MULTINET:NTPD.LOG file. (See <i>Troubleshooting Tips</i> .)
NTP.DRIFT	The MULTINET:NTP.DRIFT file consists of data maintained by NTPD and used to speed up clock frequency adjustments when NTPD is restarted. You should not modify this file.

# Configuration

## NTP Network Design

NTP does not attempt to synchronize clocks to each other. Rather, each server attempts to synchronize to Universal Coordinated Time (UTC) using the best available sources and available transmission paths to those sources. This is a fine point which is worth understanding. A group of NTP-synchronized clocks may be close to each other in time, but this is not a consequence of the clocks in the group having synchronized to each other, but rather because each clock has synchronized closely to UTC via the best source it has access to.

The most important factor in providing accurate, reliable time is the selection of modes and servers to be used in the configuration file. An NTP network should consist of a multiply redundant hierarchy of servers and clients, with each level in the hierarchy identified by stratum number. Primary servers operate at stratum one and provide synchronization to secondary servers operating at stratum two and so on to higher strata. In this hierarchy, clients are simply servers that have no dependents.

Determine which list of peers/servers you want to include in the configuration file. Include at least one (but preferably two) peer or server hosts that you are assured:

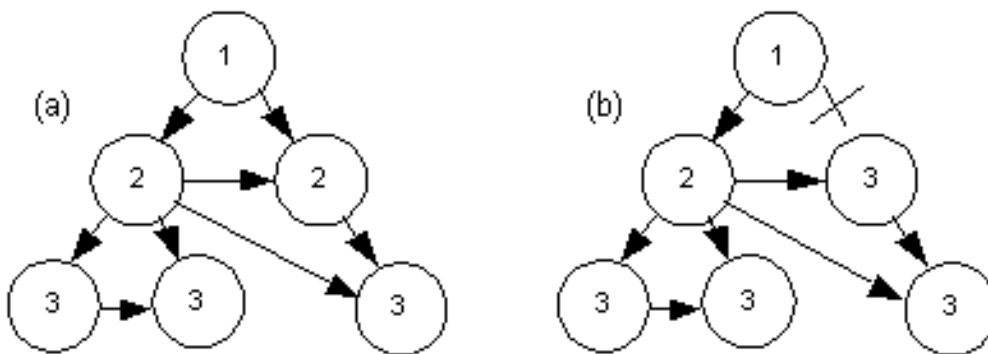
- Are running NTP
- Provide accurate time
- Synchronize to Internet Time Servers (if they are not themselves ITSs)

Two hosts provide reliability in case one goes down. You do not need to identify what stratum each host is. NTP determines this through the reference information it sends in its data exchanges.

NTP data is exchanged periodically between hosts as encapsulated in UDP datagrams, and adjustments are made based on an NTP algorithm. The frequency of exchange is related to the server's experience of time corrections. The more accurate the local clock becomes over time and after many adjustments, the less often the NTP server checks the need for corrections. The frequency of exchange is rarely intrusive to normal network operation. Also, the unreliability of UDP has no measurable impact on the process, and the process does not depend on any such reliability.

Primary servers are servers with reliable time sources, such as GPS receivers or atomic clocks, and can be found on the public internet, or set up within an intranet. Stratum numbers equate to the number of intermediate servers (or *hops*) between a given host and the Stratum 1 server it is ultimately referencing. Stratum numbers are not assigned statically, but change as server connections change. NTP servers can be (and often are) other types of systems running NTP, not just OpenVMS systems.

The stratum method allows for backup timekeeping in case a node or connection goes down, and stratum numbers may change as a consequence. In diagram A below, each node has a stratum number based on hop count, with the ITS at the top of the pyramid. The solid arrows are the active synchronization paths and direction of timing information flow; the lighter arrows are background synchronization paths where timing information is exchanged but not necessarily used for synchronization. Diagram B below shows the same network with one of the connections broken — note that the stratum for the affected peer increases from 2 to 3.



NTP makes local system time adjustments by either *slewing* or *stepping* the clock. Slewing runs the clock faster or slower than its normal frequency. Stepping sets the clock immediately to the correct time. Stepping occurs infrequently, only when there is a large time offset to adjust, such as when starting NTPD or when making daylight savings time (DST) changes.

Under some circumstances it can be disruptive to step the clock, such as when running database software that journals transactions. Such software can become very confused when a transaction is completed prior to the time at which it began, such as can happen when a clock is stepped backwards during a transaction's lifetime. In such cases the `slewalways` configuration option can be used to turn off stepping of the clock, and force all adjustments to be made by slewing. For large time changes,

such as DST changeovers, the adjustment can take a long time (several hours) to complete, and during this time the system's time will not be correct. For this reason it is not wise to allow a system set for `slewalways` to act as a server to another system.

In determining your NTP network design, keep in mind the way that the NTP protocol works, and how NTPD will determine the correct time. There should be several time servers in the configuration for each node, with good, reliable and non-congested paths between them. Nodes that will act only as clients can use the `slewalways` option, but nodes used as time sources by other nodes should generally allow stepping of the time so that inaccurate times are not reported for extended periods at Daylight Savings Time (DST) changeovers. See the *NTP.CONF* section of this chapter for more information on the `slewalways` option and the *Timezone* section for more information on DST handling.

## Authentication

NTP implements a general purpose address- and mask-based restriction list (see the `restrict` configuration option). While this is not adequate to prevent hacking attacks, it can be useful to lock out a malfunctioning server that is disrupting normal operations. See the *Access Control Commands* section for more information.

The NTP standard specifies an extension which provides cryptographic authentication of received NTP packets. This is implemented in NTPD using the MD5 algorithm to compute a digital signature, or message digest. See the *Authentication Using a Keys File* section for more information.

## Finding Servers

In many large organizations there is an administrator for the organization's networks which handles management of various network services. This person or department can usually provide information on local NTP servers, and often suggest configurations that are known to work.

There are also a number of publicly available time servers on the internet. A list of such servers can be accessed via the web at <http://www.eecis.udel.edu/~ntp> These data are updated on a regular basis using information provided voluntarily by various site administrators.

## NTP.CONF

The `NTP.CONF` file is used to specify the initial configuration of the NTPD server. It contains information about servers and peers, modes of operation, non-default file names, and other configuration data. See the table for a list of available options, and a brief description of what each does.

peer	<pre>peer [ address ] [ version 4 ] [key 0] [minpoll 6] [maxpoll 10]</pre> <p>Specifies that the server is to operate in symmetric active mode with the specified remote server. In this mode, the local server can be synchronized to the remote server and, in addition, the remote server can be synchronized by the local server. This is useful in a network of servers where, depending on various failure scenarios, either the local or remote server may be the better source of time.</p> <p>The <code>address</code> can be a domain name or an IP address in dotted quad notation</p> <p>The <code>key</code> specifies that all packets sent to an address are to include authentication fields encrypted using the specified key identifier, which is an unsigned 32-bit integer. The default is to not include an encryption field.</p> <p><code>version</code> specifies the protocol version number to be used for outgoing NTP packets. Versions 1, 2, 3 and 4 are the choices, with version 4 the default.</p>
server	<pre>server [ address ] [ version 4 ] [key 0] [minpoll 6] [maxpoll 10]</pre> <p>Specifies that the local server is to operate in client mode with the specified remote server. In this mode, the local server can be synchronized to the remote server, but the remote server can never be synchronized to the local server.</p>

	<p>The <i>address</i> can be a domain name or an IP address in dotted quad notation</p> <p>The <i>key</i> specifies that all packets sent to an address are to include authentication fields encrypted using the specified key identifier, which is an unsigned 32-bit integer. The default is to not include an encryption field.</p> <p><i>version</i> specifies the protocol version number to be used for outgoing NTP packets. Versions 1, 2, 3 and 4 are the choices, with version 4 the default.</p>
broadcast	<pre>broadcast [ address ] [ version 4 ] [ key 0 ] [ ttl 1 ]</pre> <p>Specifies broadcast mode, where the local server sends periodic broadcast messages to a client population at the broadcast/multicast address specified. This specification applies only to the local server operating as a sender. For operation as a broadcast client, see the <code>broadcastclient</code> option that follows. In this mode <i>address</i> is usually the broadcast address on (one of) the local networks.</p> <p>The <i>key</i> specifies that all packets sent to an address are to include authentication fields encrypted using the specified key identifier, which is an unsigned 32-bit integer. The default is to not include an encryption field.</p> <p><i>version</i> specifies the protocol version number to be used for outgoing NTP packets. Versions 1, 2, 3 and 4 are the choices, with version 4 the default.</p> <p><i>ttl</i> specifies the number of routers to pass through before the packet is discarded. The default is 127 routers.</p>

<p>broadcastclient</p>	<p>broadcastclient</p> <p>This command directs the local server to listen for broadcast messages at the broadcast address of the local network. Upon hearing a broadcast message for the first time, the local server measures the nominal network delay using a brief client/server exchange with the remote server, then enters the <code>broadcastclient</code> mode, in which it listens for and synchronizes to succeeding broadcast messages.</p> <div style="background-color: #e6f2ff; padding: 10px; border-left: 3px solid #0070c0;"> <p><b>Note:</b> In order to avoid accidental or malicious disruption in this mode, both the local and remote servers should operate using authentication and the same trusted key and key identifier.</p> </div>
<p>local-master</p>	<p>local-master <i>stratum</i></p> <p>Indicates that the system is to act as an authoritative time source for other systems at a stratum level lower than the specified <i>stratum</i>.</p>
<p>multicastclient</p>	<p>multicastclient [ address ]</p> <p>Specifies that this host is a multicast client for multicasts to the specified multicast address.</p>
<p>manycastclient</p>	<p>manycastclient [address] [version 4] [key 0] [minpoll 6] [maxpoll 10]</p> <p>Specify that this host is a manycast client and provide relevant settings.</p>
<p>manycastserver</p>	<p>manycastserver [ address ]</p> <p>Specify that this host is a manycast server for the given address.</p>
<p>broadcastdelay</p>	<p>broadcastdelay 0.004</p> <p>The broadcast and multicast modes require a special calibration to determine the network delay between the local and remote servers. Ordinarily, this is done automatically by the initial protocol exchanges</p>

	<p>between the client and server. In some cases, the calibration procedure may fail due to network or server access controls, for example. This command specifies the default delay to be used under these circumstances. Typically (for Ethernet), a number between 0.003 and 0.007 seconds is appropriate. The default when this command is not used is 0.004 seconds.</p>
<p>restrict</p>	<pre>restrict [ address ] [ mask 255.255.255.0 ] ignore noserve notrust noquery</pre> <p>Restrict access from and to the specified address for the specified types of access.</p> <p>The <code>address</code> argument, expressed in dotted quad form, is the address of a host or network. The <code>mask</code> argument, also expressed in dotted quad form, defaults to 255.255.255.255, meaning that the <code>address</code> 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.</p> <p><b>Note!</b> While <code>numeric-address</code> is normally given in dotted-quad format, the text string default, with no mask option, can be used to indicate the default entry.</p> <p><code>ignore</code> - Ignores all packets from hosts which match this entry. If this flag is specified, neither queries nor time server polls are responded to.</p> <p><code>noquery</code> - ignores all NTP mode 6 and 7 packets (information queries and configuration requests generated by NTPQ and NTPDC) from the source. Time service is not affected.</p> <p><code>noserve</code> - Ignores NTP packets whose mode is other than 6 or 7. In effect, time service is denied, though queries may still be permitted.</p>

	<p><code>notrust</code> - Treats these hosts normally in other respects, but never uses them as synchronization sources.</p>
<code>driftfile</code>	<p><code>driftfile file_name</code></p> <p>Specify the name of the drift file. The default is <code>MULTINET:NTP.DRIIFT</code> if this option is not used.</p> <p>The drift file is used to record the frequency offset of the local clock oscillator. If the file exists, it is read at startup in order to set the initial frequency offset and then updated once per hour with the current frequency offset computed by the daemon. If the file does not exist or this command is not given, the initial frequency offset is assumed zero. In this case, it may take some hours for the frequency to stabilize and the residual timing errors to subside.</p>
<code>keys</code>	<p><code>keys file_name</code></p> <p>Specify the name of the keys file. The default is <code>MULTINET:NTP.KEYS</code> if this option is not used.</p>
<code>statsdir</code>	<p><code>statsdir path</code></p> <p>Indicates the full path of a directory where statistics files should be created. This keyword allows the (otherwise constant) <code>filegen</code> filename prefix to be modified for file generation sets, which is useful for handling statistics logs.</p>
<code>filegen</code>	<p><code>filegen [ file filename ] [ type typename ] [ enable   disable ]</code></p> <p>Configures the generation fileset name. Generation filesets provide a means for handling files that are continuously growing during the lifetime of a server. Server statistics are a typical example for such files.</p>



At most one element of the set is being written to at any one time. The type given specifies when and how data is directed to a new element of the set.

`filename` - This string is directly concatenated to the directory `MULTINET:` or the directory prefix specified using the `statsdir` option. The suffix for this filename is generated according to the type of a fileset.

`typename` - A file generation set is characterized the following `typenames`:

- `none` - One element of the fileset is used for each, NTPD server.
- `day` - One file generation set element is created per day. A day is defined as the period between 00:00 and 24:00 UTC. The fileset member suffix consists of a dot (.) and a day specification in the form `YYYYMMDD`. `YYYY` is a 4-digit year number (such as 2003). `MM` is a two digit month number. `DD` is a two digit day number. Thus, all information written at 10 December 2002 would end up in a file named `prefix filename.20021210`.
- `week` - Any fileset member contains data related to a certain week of a year. The term week is defined by computing day-of-year modulo 7. Elements of such a file generation set are distinguished by appending the following suffix to the fileset filename base: a dot, a 4-digit year number, the letter `w`, and a 2-digit week number. For example, information from January 10th, 2003 would end up in a file with suffix `.2003W1`.
- `month` - One generation fileset element is generated per month. The filename suffix consists of a dot, a 4-digit year number, and a 2-digit month.
- `year` - One generation file element is generated per year. The filename suffix consists of a dot and a 4-digit year number.
- `age` - This type of file generation sets changes to a new element of the fileset every 24 hours of server operation. The filename suffix consists of a dot, the letter `a`, and an 8-digit number. This number is taken to be the number of seconds the server is running at the start of the corresponding 24-hour period.

Information is only written to a file generation by specifying `enable`; output is prevented by specifying `disable`.

publickey	<p>publickey file_name</p> <p>Specify the public keys file location.</p>
privatekey	<p>privatekey file_name</p> <p>Specify the private key file location.</p>
clientlimit	<p>clientlimit [ n ]</p> <p>Sets the client_limit variable that limits the number of simultaneous access-controlled clients. The default value is 3.</p>
clientperiod	<p>clientperiod [ 3600 ]</p> <p>Sets the client_limit_period variable that specifies the number of seconds after which a client is considered inactive and thus no longer is counted for client limit restriction. The default value is 3600 seconds.</p>
trustedkey	<p>trustedkey [key]</p> <p>Specifies the encryption key identifiers which are trusted for the purposes of authenticating peers suitable for synchronization. The authentication procedures require that both the local and remote servers share the same key and key identifier for this purpose, although different keys can be used with different servers. The key arguments are 32-bit unsigned integers.</p> <div style="background-color: #e6f2ff; padding: 10px; margin-top: 20px;"> <p><b>Note:</b> NTP key 0 is fixed and globally known. If meaningful authentication is to be performed, the 0 key should not be trusted.</p> </div>
requestkey	<p>requestkey [ key]</p> <p>Specifies the key identifier to use with the NTPDC program, which uses a proprietary protocol specific to this distribution of NTPD. The key</p>

	<p>argument to this command is a 32-bit unsigned integer. If no <code>requestkey</code> command is included in the configuration file, or if the keys do not match, NTPDC requests are ignored.</p>
<code>controlkey</code>	<p><code>controlkey [ key ]</code></p> <p>Specifies the key identifier to use with the NTPQ program, which uses the standard protocol defined in RFC 1305. The key argument to this command is a 32-bit unsigned integer. If no <code>controlkey</code> command is included in the configuration file, or if the keys do not match, NTPQ requests are ignored.</p>
<code>setvar</code>	<p><code>setvar [ value ]</code></p> <p>This command adds an additional system variable. These variables can be used to distribute additional information such as the access policy. If the variable of the form <code>name = value</code> is followed by the <code>default</code> keyword, the variable is listed as part of the default system variables (<code>ntpq rv</code> command). These additional variables serve informational purposes only. They are not related to the protocol other than that they can be listed. The known protocol variables always override any variables defined using the <code>setvar</code> mechanism.</p>
<code>logfile</code>	<p><code>logfile file_name</code></p> <p>Specify the logfile name. The default is <code>MULTINET:NTPD.LOG</code> if this option is not specified.</p>
<code>logconfig</code>	<p><code>logconfig [+ - =]</code>  <code>[{sync sys peer clock}{{,all}}{info statistics events status}}]...</code></p> <p>Specify logging options.</p>
<code>enable</code>	<p><code>enable auth bclient ntp kernel monitor stats calibrate</code></p> <p>Enable various options.</p>

	<p><code>auth</code> - Enables the server to synchronize with unconfigured peers only if the peer was correctly authenticated using a trusted key and key identifier. The default for this setting is <code>disable</code>.</p> <p><code>bclient</code> - When enabled, this is identical to the <code>broadcastclient</code> command. The default for this flag is <code>disable</code>.</p> <p><code>ntp</code> - 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 flag 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. The default for this flag is <code>enable</code>.</p> <p><code>kernel</code> - this setting is not used in the MultiNet implementation.</p> <p><code>monitor</code> - Enables the monitoring facility. See the <code>monlist</code> command of the NTPDC program for further information. The default for this flag is <code>enable</code>.</p> <p><code>stats</code> - Enables the statistics facility. For further information, see <i>Monitoring Commands</i>. The default for this flag is <code>enable</code>.</p> <p><code>calibrate</code> - this setting is not used in the MultiNet implementation.</p>
<p><code>disable</code></p>	<p><code>disable</code>  <code>auth bclient ntp kernel monitor stats calibrate</code></p> <p>Disable various options. See the <code>enable</code> entry for details.</p>
<p><code>slewalways</code></p>	<p><code>Slewalways</code></p> <p>Specify that the clock time is always to be slewed, never stepped.</p>

	<p>NTPD normally steps the clock when there is a relatively large time error to adjust. The <code>slewalways</code> command directs the local NTP server to always slew the clock, regardless of how large the required correction is. This command is useful to avoid an abrupt one hour clock change when daylight savings time (DST) changes occur. For DST changes when <code>slewalways</code> is specified, NTPD slews the clock over a period of about 6 hours.</p>
panic	<p><code>panic max_adjust_time</code></p> <p>Set the maximum time change that will be allowed. If non-zero, this value should always be at least 4000 seconds to allow for DST time changes even on systems with large time errors.</p> <p>If set to zero, the panic sanity check is disabled and a clock offset of any value will be accepted.</p>
debug	<p><code>debug [level]</code></p> <p>Set the debug logging severity level.</p>
set_vms_logicals	<p><code>set_vms_logicals</code></p> <p>Causes the NTP server to also adjust the value of the VMS logicals <code>SYS\$TIMEZONE_DIFFERENTIAL</code>, <code>SYS\$TIMEZONE_DAYLIGHT_SAVING</code> and <code>SYS\$TIMEZONE_NAME</code> when it changes the <code>MULTINET_TIMEZONE</code> logical at DST start or end. The following files will also be updated:  <code>DTSS\$TIMEZONE_DIFFERENTIAL</code> and <code>SYS\$TIMEZONE.DAT</code></p> <p>NTP does NOT set <code>SYS\$TIMEZONE_RULE</code>, which generally does not change. The format of <code>SYS\$TIMEZONE_RULE</code> is specified in <code>SYS\$MANAGER:UTC\$TIME_SETUP.COM</code>.</p>
call_dst_proc	<p><code>call_dst_proc</code></p>

	Causes the NTP server to spawn a subprocess to execute the <code>MULTINET:NTPD_DST_PROC.COM</code> procedure, if such a procedure file exists with the proper protections, when changing into or out of DST, or when first starting up. See the <i>Using the call_dst_procoption</i> below for more information.
<code>set_clock_daily</code>	<p><code>set_clock_daily</code></p> <p>When this is included in <code>NTP.CONF</code>, then NTPD will only make one call a day (instead of once an hour) to the routine that sets the TOY clock to ensure that the value is preserved. This reduces the number of entries in the Integrity system event logs. NTPD may set the clock more often than daily, but it will be done only to correct any drift that is detected. In our tests on a RX2600 this happened approximately every 6 hours.</p>

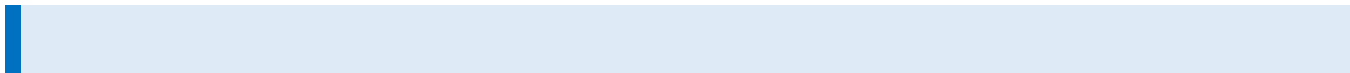
# Time Zone Configuration and Hardware Clock Overview

By OpenVMS convention, the system clock is usually set to the local time, but network protocols represent time in Coordinated Universal Time (UTC, sometimes referred to as GMT). To convert between local time and UTC, MultiNet uses built-in rules or rules provided by the system manager.

Each country or geographical area has its own names for time zones and its own rules for Daylight Savings Time (DST). The names of these time zones and rules are not necessarily unique. For example, "EST" could refer to the United States Eastern Standard Time, the Canadian Eastern Standard Time (which uses different DST rules), or the Australian Eastern Standard Time (which is a different offset from UTC as well as having different DST rules).

MultiNet uses the name of the local time zone as specified by the system manager, along with time zone rules, to calculate the offset between the local time and UTC, so it is important that an appropriate set of time zone rules be selected for the location where the system is located.

MultiNet assumes that the hardware clock is always set exactly to local time. For a smooth transition to and from Daylight Savings Time the hardware clock must be reset at the appropriate time.



**Note:** Using a military time zone or an explicit GMT offset disables automatic Daylight Savings Time transitions.

# Time Zone Support

Because it is impossible to anticipate every country or area in which MultiNet might be used, and because the Daylight Savings Time rules are subject to change by government action, MultiNet permits you to write your own site-specific time zone rules. There are two types of time zone rules: compiled-in and loadable.

- Compiled-in rules are geographically centered around the United States but also include foreign time zones whose names do not conflict with the U.S. time zones.
- Loadable rules are selected with the `MULTINET NETCONFIG SET TIMEZONE-RULES` command and can be used to override the compiled-in rules.

MultiNet includes a database of the most common loadable rules. You can select these rules as-is, or modify them to conform to the correct local time zone rules.

When MultiNet searches the time zone rules looking for a zone, it first searches the loadable rules in the order they are specified, then searches the compiled-in rules. This method allows you to change the compiled-in rules by loading rules that override them.

In addition to the standard one-letter U.S. military time zones and time zones of the form `GMT+hh:mm` or `GMT-hh:mm`, there are compiled-in time zone rules supported by MultiNet which are shown in the below table.

Time Zone Name	GMT Offset	DST Rules	Area or Country
EST or EDT	-5 hours	U.S. Federal	Eastern United States
CST or CDT	-6 hours	U.S. Federal	Central United States
MST or MDT	-7 hours	U.S. Federal	Mountain United States

PST or PDT	-8 hours	U.S. Federal	Pacific United States
YST or YDT	-9 hours	U.S. Federal	Yukon
HST	-10	-none-	Hawaii
NST or NDT	-3:30 hours	Canadian	Canadian Newfoundland
AST or ADT	-4 hours	Canadian	Canadian Atlantic
JST	+9 hours	-none-	Japan
SST	+8 hours	-none-	Singapore
GMT	+0 hours	-none-	Greenwich Mean Time
GMT or BST	+0 hours	British	Britain
WET or WET-DST	+0 hours	European	Western Europe
MET or MET-DST	+1 hour	European	Middle Europe
CET or CET-DST	+1 hour	European	Central Europe (Middle Europe)
EET or EET-DST	+2 hours	European	Eastern Europe
NZST or NZDT	+12 hours	New Zealand	New Zealand

## Loadable Time Zone Rules

Loadable time zone rules provided with MultiNet are in the text file `MULTINET:TIMEZONES.DAT`. You can copy this file to `MULTINET:TIMEZONES.LOCAL`, and then add user-written time zone rules to override the compiled-in rules.

Loadable time zone rules consist of three parts:



COUNTRY	A collection of time zones (ZONES). For example, the country US selects all U.S. time zones. This provides a convenient way to select groups of time zones.
ZONE	A specification of a particular time zone, including the name of the zone, the UTC offset, the DST rules in effect, and the name to use while DST is in effect.
RULE	A rule for determining when DST is in effect.

## Format of COUNTRY Specification

```
COUNTRY countryname zonename [zonename . . .]
```

The COUNTRY specification gives the name of a geographical area and the names of the time zones associated with it. This provides a way to group time zones so they may be selected more conveniently.

The following example shows the definition of the country "US", listing the zones corresponding to the United States. The example for Arizona is slightly different, showing the zone "US/Arizona" instead of "US/Mountain." ("US/Arizona" is the definition of a Mountain time zone that does not observe Daylight Savings Time.)

```
Country US  US/Eastern US/Central US/Mountain US/Pacific US/Yukon US/Hawaii
Country US/Arizona -
US/Eastern US/Central US/Arizona US/Pacific US/Yukon US/Hawaii
```

## Format of ZONE Specification

```
ZONE zonename gmtoffset rulename standard-name dst-name [COMPILED_IN]
```

zonename	The name by which this zone can be selected, or the name by which it is referred to in a COUNTRY specification.
gmtoffset	This zone's standard time offset from UTC.
rulename	Is the name of the RULE specification that determines when DST is in effect for this zone. The rulename may be an underscore (_) to indicate that this zone does not use DST.

standard-name and dst-name	The names by which this zone is referred to during standard time, and during Daylight Savings Time, respectively. These are the names by which SET TIMEZONE selects the local time zone.
-------------------------------	--

The ZONE specification describes a time zone:

If there are no DST rules, the `dst-name` should be specified as an underscore (`_`). The optional `COMPILED_IN` keyword indicates that this rule is compiled-in and need not be loaded, as long as no other rules conflict with it. If you edit a `COMPILED_IN` ZONE specification, you must remove the `COMPILED_IN` keyword to force the ZONE specification to be loaded.

The following example shows the definition of the normal United States Mountain time zone. The Arizona example shows the definition of a Mountain time zone that does not observe Daylight Savings Time.

```
Zone US/Mountain -7:00 US MST MDT COMPILED_IN
Zone US/Arizona -7:00 _ MST
```

## Format of a RULE Specification

```
RULE rulename startyear ruletype save start-date end-date
```

The RULE specification describes a rule or set of rules for determining at what times DST is in effect:

rulename	The name of the RULE specification in ZONE specifications.
startyear	The year during which this DST rule takes effect. The rule remains in effect until a later <code>startyear</code> is specified in a rule with the name <code>rulename</code> .
ruletype	Specifies the type of DST rules. There are three permitted values: <ul style="list-style-type: none"> <li>• <code>DST</code> indicates normal Northern-Hemisphere Daylight Savings Time rules, which switch at the time and date indicated.</li> <li>• <code>REV_DST</code> indicates normal Southern-Hemisphere Daylight Savings Time rules.</li> <li>• <code>NULL</code> indicates that no Daylight Savings Time is in effect during the specified years.</li> </ul>
save	Indicates the difference between Standard Time and DST.

start-date and end-date	Specify the starting and ending dates for DST. Specific dates can be specified, or rules such as "First Sunday" or "Last Sunday" can be used. For days other than the "First" or "Last" you must use <code>dayname &gt;= date</code> . For example the second Sunday is expressed as <code>Sunday &gt;= 8</code> . See the file <code>MULTINET:TIMEZONES.DAT</code> for examples of specifying dates.
-------------------------------	---

The following example illustrates the United States Federal Daylight Savings Time rules:

```
Rule US 2007 DST 1:00 Sunday>=8 March 2:00 First Sunday November 2:00
Rule US 1987 DST 1:00 First Sunday April 2:00 Last Sunday October 2:00
Rule US 1976 DST 1:00 Last Sunday April 2:00 Last Sunday October 2:00
Rule US 1975 DST 1:00 23 February 2:00 Last Sunday October 2:00
Rule US 1974 DST 1:00 6 January 2:00 Last Sunday October 2:00
Rule US 1970 DST 1:00 Last Sunday April 2:00 Last Sunday October 2:00
```

## Loadable Time Zone Rules Provided with MultiNet

This table shows the loadable rules provided in the `MULTINET:TIMEZONES.DAT` file which you may modify or augment as appropriate for your location.

Country Name	Rule Name	Time Zone Name	GMT Offset	DST Rules
	GMT	GMT <sup>2</sup>	0 hours	-none-
	UT	UT <sup>a</sup>	0 hours	-none-
US-Military	US-Military/Z <sup>a</sup>	Z	0 hours	-none-
US-Military	US-Military/A <sup>a</sup>	A	-1 hour	-none-
US-Military	US-Military/B <sup>a</sup>	B	-2 hours	-none-
US-Military	US-Military/C <sup>a</sup>	C	-3 hours	-none-
US-Military	US-Military/D <sup>a</sup>	D	-4 hours	-none-

<sup>2</sup> This timezone is compiled-in also.

US-Military	US-Military/E <sup>a</sup>	E	-5 hours	-none-
US-Military	US-Military/F <sup>a</sup>	F	-6 hours	-none-
US-Military	US-Military/G <sup>a</sup>	G	-7 hours	-none-
US-Military	US-Military/H <sup>a</sup>	H	-8 hours	-none-
US-Military	US-Military/I <sup>a</sup>	I	-9 hours	-none-
US-Military	US-Military/K <sup>a</sup>	K	-10 hours	-none-
US-Military	US-Military/L <sup>a</sup>	L	-11 hours	-none-
US-Military	US-Military/M <sup>a</sup>	M	-12 hours	-none-
US-Military	US-Military/N <sup>a</sup>	N	1 hour	-none-
US-Military	US-Military/O <sup>a</sup>	O	2 hours	-none-
US-Military	US-Military/P <sup>a</sup>	P	3 hours	-none-
US-Military	US-Military/Q <sup>a</sup>	Q	4 hours	-none-
US-Military	US-Military/R <sup>a</sup>	R	5 hours	-none-
US-Military	US-Military/S <sup>a</sup>	S	6 hours	-none-
US-Military	US-Military/T <sup>a</sup>	T	7 hours	-none-
US-Military	US-Military/U <sup>a</sup>	U	8 hours	-none-
US-Military	US-Military/V <sup>a</sup>	V	9 hours	-none-
US-Military	US-Military/W <sup>a</sup>	W	10 hours	-none-
US-Military	US-Military/X <sup>a</sup>	X	11 hours	-none-

US-Military	US-Military/Y <sup>a</sup>	Y	12 hours	-none-
US	US/Eastern <sup>a</sup>	EST/EDT	-5 hours	US Federal
US	US/Central <sup>a</sup>	CST/CDT	-6 hours	US Federal
US	US/Mountain <sup>a</sup>	MST/MDT	-7 hours	US Federal
US	US/Pacific <sup>a</sup>	PST/PDT	-8 hours	US Federal
US	US/Yukon <sup>a</sup>	YST/YDT	-9 hours	US Federal
US	US/Hawaii <sup>a</sup>	HST	-10 hours	-none-
US/East-Indiana	US/East-Indiana	EST	-5 hours	-none-
US/Arizona	US/Arizona	MST	-7 hours	-none-
Canada	Canada/Newfoundland <sup>a</sup>	NST/NDT	-3:30 hours	Canadian
Canada	Canada/Atlantic <sup>a</sup>	AST/ADT	-4 hours	Canadian
Canada	Canada/Eastern	EST/EDT	-5 hours	Canadian
Canada	Canada/Central	CST/CDT	-6 hours	Canadian
Canada	Canada/Mountain	MST/MDT	-7 hours	Canadian
Canada	Canada/Pacific	PST/PDT	-8 hours	Canadian
Canada	Canada/Yukon	YST/YDT	-9 hours	Canadian
Canada	Canada/Saskatchewan	CST	-6 hours	-none-
Israel	Israel	IST/DST	+2 hours	Israeli
Australia	Australia/Tasmania	EST	10 hours	Australian

Australia	Australia/Queensland	EST	10 hours	-none-
Australia	Australia/North	CST	9:30 hours	-none-
Australia	Australia/West	WST	8 hours	-none-
Australia	Australia/South	CST	9:30 hours	Australian
Australia	Australia/Victoria	CST	10 hours	Australian
Australia	Australia/NSW	CST	10 hours	Australian
Australia	Australia/Yarcowinna	CST	9:30 hours	Australian
Australia	Australia/LHI	CST	10:30 hours	Australian
Europe	Britain <sup>a</sup>	GMT/BST	0 hours	GB-Eire
Europe	Europe/Western <sup>a</sup>	WET/WET-DST	0 hours	W-Eur
Europe	Europe/Middle <sup>a</sup>	MET/MET-DST	1 hour	M-Eur
Europe	Europe/Central <sup>a</sup>	CET/CET-DST	1 hour	M-Eur
Europe	Europe/Eastern <sup>a</sup>	EET/EET-DST	2 hours	E-Eur
	Iceland	GMT	1 hour	-none-
	Poland	MET	1 hour	W-Eur
	Turkey	EET/EET/DST	3 hours	Turkey
Japan	Japan <sup>a</sup>	JST	+9 hours	-none-
Singapore	Singapore <sup>a</sup>	SST	+8 hours	-none-
New Zealand	New Zealand <sup>a</sup>	NZST/NZDT	+12 hours	New Zealand

## Selecting Time Zone Rules

Time zone rules and the local time zone name are set in the MultiNet system startup command file, `MULTINET:START_MULTINET.COM`. You can use the MultiNet `NET-CONFIG` utility to specify the local time zone and which rules to load using the `SET TIMEZONE` and `SET TIMEZONE-RULES` commands. The following example shows how to select the United States Arizona rules and the local time zone MST:

```
$ MULTINET CONFIGURE  
NET-CONFIG>SET TIMEZONE MST  
NET-CONFIG>SET TIMEZONE-RULES US/ARIZONA  
NET-CONFIG>EXIT
```

## Using the `call_dst_proc` option

When NTPD is started, and whenever the local time zone shifts between daylight savings DST and standard (STD) time, if the local zone rule specifies such behavior, the NTPD server will check the `MULTINET_TIMEZONE` logical, and set it if required. The setting will only be between the DST name and the STD name for the zone, so the configuration described above is still necessary, but if your system was down during a DST shift, this can correct the logical name to match the current system clock time and the applicable zone rule when NTPD is started. If your `NTP.CONF` file specifies the `set_vms_logicals` option, the `SYS$TIMEZONE_DIFFERENTIAL`, `SYS$TIMEZONE_DAYLIGHT_SAVING` and `SYS$TIMEZONE_NAME` logicals will be updated as well.

Since there are many systems with other time-related logical names, or other items that may need updating or adjusting based on a DST change, the `call_dst_proc` option has been provided. If this option is used in `NTP.CONF`, the NTPD server will look for a file called `MULTINET:NTPD_DST_PROC.COM` any time it checks on the `MULTINET_TIMEZONE` logical (at startup and at a DST shift). If this file exists, and has the proper protections (no `WORLD` write or execute access, and owned by `SYSTEM([1,4])`) a sub-process will be spawned to execute it. This procedure can contain any commands needed, but care should be exercised in constructing this file, as it will be executing with the same privileges as the NTPD process. A “placeholder” procedure is included with MultiNet, but its contents are all comments and will do nothing as shipped.

The invocation of the `MULTINET:NTPD_DST_PROC.COM` procedure will be equivalent to this:

```
@MULTINET:NTPD_DST_PROC.COM p1 p2 p3 p4 p5
```

Where:

- p1 = Current time zone name - string (e.g. "EST" or "EDT")
- p2 = Time zone offset in seconds - integer (e.g. "-18000" or "-14400")
- p3 = DST in effect? - boolean ("Y", "N")
- p4 = In Twilight Zone? - boolean ("Y", "N")
- p5 = Startup or DST change? - string ("START" or "DST")

P1, the Current Time Zone Name, is a string specifying the current name of the local time zone. For North American Eastern Standard Time, this will be “EST” in the winter, and “EDT” in the summer, when DST is active. For time zones that don’t do DST, it will always be the zone name.

P2, the Time Zone Offset, is a signed integer specifying the offset, in seconds, from UTC for the local zone, at the current time. For North American Eastern Standard Time this is “-18000” (-5 hours), for the same zone with DST in effect it is “-14400” (-4 hours).

P3, the DST flag. This will be “Y” if DST is currently in effect for the zone, and “N” if it isn’t, or if the zone doesn’t do DST.

P4, the Twilight Zone flag. When a zone exits from DST, it sets its time back an hour. This means that for that hour, the time *appears* to be a DST time by the local DST rules, but isn’t really, since DST has already ended. That hour is called the “twilight zone” by MultiNet NTP. If the current time is in that period, the P4 parameter will be “Y”, otherwise it will be “N”.

P5, the startup/DST flag. This tells the procedure whether it is being called as a part of NTPD’s startup processing, or as part of a DST change.

These parameters are provided so that the procedure can take different action under different conditions. They may all be ignored if that is appropriate. The NTPD server doesn’t depend on any particular behavior, so long as the `MULTINET_TIMEZONE` logical is left alone and the system clock is not altered. The final completion status of the called procedure will be logged by the NTPD server, along with the PID of the spawned sub-process.

## Access Control Commands

NTP implements a general-purpose address- and mask-based restriction list (see the `restrict` configuration option). The list is sorted by address and by mask, and the list is searched in this order for matches, with the last match found defining the restriction flags associated with the incoming packets. The source address of incoming packets is used for the match, with the 32-bit address combined with the



mask associated with the restriction entry and then compared with the entry's address (which was also combined with the mask) to look for a match.

The restriction facility was implemented to conform with the access policies for the original NSFnet backbone time servers. While this facility may be otherwise useful for keeping unwanted or broken remote time servers from affecting your own, it should not be considered an alternative to the standard NTP authentication facility. Source address-based restrictions are easily circumvented by a determined hacker.

## Authentication Using a Keys File

The NTP standard specifies an extension which provides cryptographic authentication of received NTP packets. This is implemented in NTPD using the MD5 algorithm to compute a digital signature, or message digest. The specification allows any one of possibly four billion keys, numbered with 32-bit key identifiers, to be used to authenticate an association. 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 the file `MULTINET:NTP.KEYS`, which should be exchanged and stored using secure procedures. There are three classes of keys involved in the current implementation. One class is used for ordinary NTP associations, another for the NTPQ utility program, and the third for the NTPDC utility program.

### Key File Format

For MD5, keys are 64 bits (8 bytes), read from the `MULTINET:NTP.KEYS` file. While key number 0 is fixed by the NTP standard (as 64 zero bits) and may not be changed, one or more of the keys numbered 1 through 15 may be arbitrarily set in the keys file.

The keys file uses the same comment conventions as the configuration file. Key entries use a fixed format of the form:

```
keyno type key
```

- `keyno` is a positive integer
- `type` is a single character `M` for the MD5 key format
- `key` is the key itself

The key is a one to eight character ASCII string using the MD5 authentication scheme.

**Note:** Both the keys and the authentication scheme must be identical between a set of peers sharing the same key number.

**Note:** The keys used by the NTPQ and NTPDC programs are checked against passwords requested by the programs and entered by hand.

## NTP Utilities

There are several utility programs included with NTP. These allow setting the system clock from a time server, querying and controlling NTP servers on the local system or on remote hosts, and tracing the chain of time servers back to the top stratum server being used to set the local time.

These utilities are all accessible through the MULTINET command (i.e. “MULTINET NTPDATE...”), or as DCL foreign command symbols through the use of the MULTINET:NTP\_DEFINE.COM procedure to define these commands. The same image is executed in either case and it is mostly a matter of personal preference which is used. The foreign commands can be undefined by use of the MULTINET:NTP\_UNDEFINE.COM procedure.

---

# NTPDATE

The `NTPDATE` utility sets the local date and time, by polling the NTP servers given as the server arguments, 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 of these.

**Note:** The accuracy and reliability of `NTPDATE` depends on the number of servers, the number of polls each time it is run, and the interval between runs.

The `NTPDATE` utility can be run manually as necessary to set the host clock, or it can be run from the system startup command file to set the clock at boot time. This is useful in some cases to set the clock initially before starting the NTP daemon, `NTPD`. It is also possible to run `NTPDATE` from a batch job. However, it is important to note that `NTPDATE` with contrived batch jobs is no substitute for the NTP daemon, which uses sophisticated algorithms to maximize accuracy and reliability while minimizing resource use. Finally, since `NTPDATE` does not discipline the host clock frequency as does `NTPD`, the accuracy using `NTPDATE` is limited.

The `NTPDATE` utility makes time adjustments in one of two ways. If it determines that the clock is wrong by more than 0.5 second, it simply steps the time by calling the `$SETIME` system service. If the error is less than 0.5 second, it slews the time by temporarily adjusting system clock variables. The latter technique is less disruptive and more accurate when the error is small, and works quite well when `NTPDATE` is run by a batch job every hour or two.

The `NTPDATE` utility declines to set the date if `NTPD` is running on the same host. When running `NTPDATE` every hour or two from a batch job, as an alternative to running `NTPD`, results in precise enough timekeeping to avoid stepping the clock.

## Format

```
ntpdate [ -bBdoqsuv ] [ -a key ] [ -e authdelay ] [ -k keyfile ] [ -o version ] [ -p samples ] [-t timeout ] server [ ... ]
```

## Command Line Options

### **-a *key***

Enables the authentication function and specifies the key identifier to be used for authentication as the argument *key*. The keys and key identifiers must match in both the client and server key files. The default is to disable the authentication function.

### **-B**

Force the time to always be slewed, even if the measured offset is greater than ~128 ms. The default is to step the time if the offset is greater than ~128 ms.

**Note:** If the offset is large, it can sometimes take several hours to slew the clock to the correct value. During this time, the host should not be used to synchronize clients.

### **-b**

Force the time to be stepped, rather than slewed (default). This option should be used when called from a startup file at boot time.

### **-d**

Enable the debugging mode, in which `ntpd` will go through all the steps, but not adjust the local clock. Information useful for general debugging will also be printed.

### **-e *authdelay***

Specify the processing delay to perform an authentication function as the value *authdelay*, in seconds and fraction. This number is usually small enough to be negligible for most purposes, though specifying a value may improve timekeeping on very slow CPU's.

### **-k *keyfile***

Specifies the path for the authentication key file as the string *keyfile*. The default is `MULTINET:NTP.KEYS`. This file should be in the format described for NTPD configuration.

**-o *version***

Specifies the NTP version for outgoing packets as the integer version, which can be 1, 2, 3 or 4. The default is 4. This allows NTPDATE to be used with older NTP versions.

**-p *samples***

Specifies the number of samples to be acquired from each server as the integer samples, with values from 1 to 8 inclusive. The default is 4.

**-q**

Query only - don't set the clock.

**-s**

Enables OPCOM messaging. This is designed primarily for the convenience of batch jobs.

**-t *timeout***

Specifies the maximum time waiting for a server response as the value timeout, in seconds and fraction. The value is rounded to a multiple of 0.2 seconds. The default is 1 second, a value suitable for polling across a LAN.

**-u**

Directs NTPDATE to use an unprivileged port on outgoing packets. This is most useful when behind a firewall that blocks incoming traffic to privileged ports, and you want to synchronize with hosts beyond the firewall.

**-v**

Be verbose. This option will cause ntpdate's version identification string to be logged.

---



# NTPTRACE

The NTPTRACE utility determines where a given NTP server gets its time, and follows the chain of NTP servers back to their master time source. If given no arguments, it starts with localhost. Here is an example of the output from NTPTRACE:

```
$ ntptrace
localhost: stratum 4, offset 0.0019529, synch distance 0.144135
server2ozo.com: stratum 2, offset 0.0124263, synch distance 0.115784
usndh.edu: stratum 1, offset 0.0019298, synch distance 0.011993, refid
'WWVB'
```

On each line, the fields are (left to right): the host name, host stratum, time offset between that host and the local host (as measured by NTPTRACE; this is why it is not always zero for localhost), host synchronization distance, and (only for stratum-1 servers) the reference clock ID. All times are given in seconds.

**Note:** The stratum is the server hop count to the primary source, while the synchronization distance is the estimated error relative to the primary source. The NTP server must be synchronized to a peer.

## Format

```
ntptrace [-vdn] [-r retries] [-t timeout] [server]
```

## Command Line Options

**-d**

Turns on some debugging output.

**-n**

Turns off the printing of hostnames; instead, host IP addresses are given. This may be useful if a nameserver 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**

Prints verbose information about the NTP servers.

---



# NTPDC

The NTPDC utility is used to query the NTPD server about its current state and to request changes in that state. The program runs interactively or uses command line arguments. Extensive state and statistics information is available through the NTPDC interface. In addition, nearly all the configuration options that can be specified at startup using NTPD's configuration file may also be specified at run-time using NTPDC.

The NTPDC utility uses NTP mode 7 packets to communicate with the NTP server, and can be used to query any compatible server on the network which permits it.

**Note:** Since NTP is a UDP protocol, this communication is somewhat unreliable, especially over large distances, in terms of network topology. NTPDC makes no attempt to retransmit requests, and times out requests if the remote host is not heard from within a suitable timeout time.

NTPDC's operation is specific to the NTPD implementation and can be expected to work only with this, and possibly some previous versions, of the daemon. Requests from a remote NTPDC program that affect the state of the local server must be authenticated, which requires both the remote program and local server to share a common key and key identifier.

## Command Line Format

```
ntpdc [-ilnps] [-c command] [host] [...]
```

## Command Line Arguments

(If command line arguments are omitted, NTPDC runs in interactive mode.)

**-c**

The `command` that follows is interpreted as an interactive format command and is added to the list of commands to be executed on the specified host(s). The `command` must be in double quotes if it consists of more than one word. Multiple `-c` options can be given.

**-i**

Force `ntpd` to operate in interactive mode. Prompts will be written to the standard output and commands read from the standard input.

**-l**

Obtain a list of peers which are known to the server(s). This switch is equivalent to `-c listpeers`.

**-n**

Displays all host addresses in dotted quad numeric format rather than converting them to canonical hostnames.

**-p**

Print a list of the peers known to the server as well as a summary of their state. This is equivalent to `-c peers`.

**-s**

Print a list of the peers known to the server as well as a summary of their state, but in a slightly different format than the `-p` switch. This is equivalent to `-c dmpeers`.

**host**

Sets the host to which future queries are sent, as either a hostname or a numeric address. If *host* is omitted, the local host is used.

## Interactive Commands

Interactive format commands consist of a keyword followed by zero to four arguments. Only enough characters of the full keyword to uniquely identify the command need be typed. The output of a command is normally sent to the standard output, but you can send the output of individual commands to a file by appending a greater than (>) followed by a filename to the command line.

**? [command-keyword]**

**help [command-keyword]**

A question mark (?) by itself prints a list of all the known command keywords. A question mark (?) followed by a command keyword prints function and usage information.

**delay *milliseconds***

Specifies a time interval to be added to timestamps included in requests that require authentication. This is used to enable unreliable server reconfiguration over long delay network paths or between machines whose clocks are unsynchronized.

**host [*hostname*]**

Sets the host to which future queries are sent. `Hostname` may be either a hostname or a numeric address.

**hostnames [ *yes* | *no* ]**

If `yes` is specified, host names are printed in information displays. If `no` is specified, numeric addresses are printed instead. The default is `yes`, unless modified using the command line `-n` switch.

**keyid [ *keyid* ]**

Allows a key number to be used by NTPDC to authenticate configuration requests. This must correspond to a key number the server has been configured to use for this purpose.

**quit**

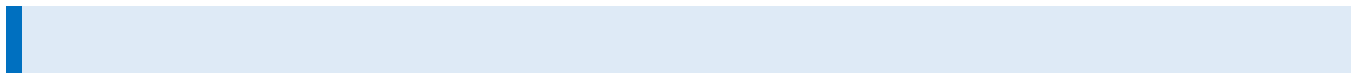
Exits NTPDC.

**passwd**

Prompts you to type in a password (which is not echoed) that is used to authenticate configuration requests. The password must correspond to the key configured for use by the NTP server for this purpose if such requests are to be successful.

**timeout [*milliseconds*]**

Specifies a timeout period for responses to server queries. The default is approximately 8000 milliseconds.



**Note:** Since NTPDC retries each query once after a timeout, the total waiting time for a timeout is twice the timeout value set.

## Control Message Commands

Query commands produce NTP mode 7 packets containing requests for information being sent to the server. These are read-only commands in that they make no modification of the server configuration state.

### **listpeers**

Obtains and prints a brief list of the peers for which the server is maintaining state. These should include all configured peer associations, as well as those peers whose stratum is such that they are considered by the server 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. Summary information includes the address of the remote peer; local interface address (0.0.0.0 if a local address has yet to be determined); stratum of the remote peer (a stratum of 16 indicates the remote peer is unsynchronized); polling interval (in seconds); reachability register (in octal); and current estimated delay, offset, and dispersion of the peer (all in seconds).

The character in the left margin indicates the mode this peer entry is operating in as per the table:

+	Symmetric active
-	Symmetric passive
=	Remote server is being polled in client mode
^	Server is broadcasting to this address
~	Remote peer is sending broadcasts
*	Peer the server is currently synchronizing to

The contents of the host field may be in one of four forms: a hostname, IP address, reference clock implementation name with its parameter, or REFCLK (implementation number, parameter). With `hostnames no`, only IP-addresses are displayed.

### **dmpeers**

A slightly different peer summary list. Identical to the output of the `peers` command, except for the character in the leftmost column. Characters only appear beside peers which were included in the final stage of the clock selection algorithm. Characters indicate server validity according to the following table:

·	peer was cast off in the <code>falseticker</code> detection
+	peer made it through <code>falseticker</code> detection
*	peer the server is currently synchronizing with

### **showpeer *peer-address* [ ... ]**

Shows a detailed display of the current peer variables for one or more peers. Most of these values are described in the NTP Version 2 specification. Understanding this information will require a detailed understanding of the inner workings of the NTP protocol, which is also available in the RFCs that specify the protocol.

### **pstats *peer-address* [ ... ]**

Shows per-peer statistic counters associated with the specified peer(s).

### **loopinfo [ *oneline* | *multiline* ]**

Prints the values of selected loop filter variables. The loop filter is the part of NTP which deals with adjusting the local system clock.

<code>loop filter</code>	is the part of NTP that deals with adjusting the local system clock
<code>offset</code>	is the last offset given to the loop filter by the packet processing code

frequency	is the frequency error of the local clock in parts per million (ppm)
time_const	controls the stiffness of the phase-lock loop and thus the speed at which it can adapt to oscillator drift
watchdog timer value	is the number of seconds 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 printed, with `multiline` as the default.

### **sysinfo**

Prints a variety of system state variables, such as the state related to the local server. All except the last four lines are described in the NTP Version 3 specification, RFC 1305.

The system flags can be set and cleared by the `enable` and `disable` configuration commands, respectively. These are the `auth`, `bclient`, `monitor`, `pll`, `pps`, and `stats` flags. (See the *NTPD* section for the meaning of these flags.)

The `stability` is the residual frequency error remaining after the system frequency correction is applied, and is intended for maintenance and debugging. In most architectures, this value initially decreases from as high as 500 ppm to a nominal value in the range .01 to 0.1 ppm. If it remains high for some time after starting the server, something might be wrong with the local clock.

The `broadcastdelay` shows the default broadcast delay, as set by the `broadcastdelay` configuration command. The `authdelay` shows the default authentication delay, as set by the `authdelay` configuration command.

### **sysstats**

Prints statistics counters maintained in the protocol module.

### **memstats**

Prints statistics counters related to memory allocation code.

### **iostats**

Prints statistics counters maintained in the input-output module.

#### **timerstats**

Prints statistics counters maintained in the timer/event queue support code.

#### **reslist**

Obtains and prints the server's restriction list. This list is usually printed in sorted order and may help to understand how the restrictions are applied.

#### **monlist [ version ]**

Obtains and prints traffic counts collected and maintained by the monitor facility. You do not normally need to specify the version number.

## **Runtime Configuration Requests**

All requests that cause state changes in the server are authenticated by the server using the `requestkey` in the configuration file (which can be disabled by the server by not configuring a key). The key number and the corresponding key must also be made known to NTPDC. This can be done using NTPDC's `keyid` and `passwd` commands, the latter of which prompts at the terminal for a password to use as the encryption key. You are also prompted automatically for both the key number and password the first time a command is given that would result in an authenticated request to the server. 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.

Authenticated requests always include a timestamp in the packet data, which is included in the computation of the authentication code. This timestamp is compared by the server to its receive timestamp. If they differ by more than a small amount, the request is rejected. This is done for two reasons. First, it makes simple replay attacks on the server, by someone who might be able to overhear traffic on your LAN, much more difficult. Secondly, it makes it more difficult to request configuration changes to your server from topologically remote hosts. While the reconfiguration facility works well with a server on the local host, and may work adequately between time synchronized hosts on the same LAN, it works very poorly for more distant hosts. As such, if reasonable passwords are chosen, care is taken in the distribution and protection of keys, and appropriate source address restrictions are applied, the run-time reconfiguration facility should provide an adequate level of security.

The following commands all make authenticated requests.

**addpeer** *peer-address* [ *keyid* ] [ *version* ] [ *prefer* ]

Adds a configured peer association at the given address and operates in symmetric active mode.

**Note:** An existing association with the same peer may be deleted when this command is executed, or may simply be converted to conform to the new configuration, as appropriate. If the optional *keyid* is a non-zero integer, all outgoing packets to the remote server have an authentication field attached, encrypted with this key. If the value is 0 (or not given), no authentication is done. The *version* can be 1, 2, 3 or 4, and defaults to 4. The *prefer* keyword indicates a preferred peer (and thus is used primarily for clock synchronization if possible).

**addserver** *peer-address* [ *keyid* ] [ *version* ] [ *prefer* ]

Identical to the `addpeer` command, except that the operating mode is client.

**broadcast** *peer-address* [ *keyid* ] [ *version* ] [ *prefer* ]

Identical to the `addpeer` command, except that the operating mode is broadcast. In this case a valid key identifier and key are required. The *peer-address* parameter can be the broadcast address of the local network, or a multicast group address assigned to NTP. If using a multicast address, a multicast-capable kernel is required.

**unconfig** *peer-address* [ ... ]

Removes the configured bit from the specified peers. In many cases, 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* ] [ ... ]

Operates the same as the `enable` and `disable` configuration file commands of NTPD.

**restrict address mask flag** [ *flag* ]

Operates the same as the `restrict` configuration file commands of NTPD.



**unrestrict address mask flag [ flag ]**

Unrestricts the matching entry from the restrict list.

**delrestrict address mask [ ntpport ]**

Deletes the matching entry from the restrict list.

**readkeys**

Causes the current set of authentication keys to be purged and a new set to be obtained by rereading the keys file (`MULTINET:NTP.KEYS`). This allows encryption keys to be changed without restarting the server.

**trustedkey keyid [ ... ]**

**untrustedkey keyid [ ... ]**

Operates the same as the `trustedkey` and `untrustedkey` configuration file commands of NTPD.

**authinfo**

Returns information concerning the authentication module, including known keys and counts of encryptions and decryptions which have been done.

**reset**

Clears the statistics counters in various modules of the server.

---

# NTPQ

The `NTPQ` utility is used to query NTP servers that implement the recommended NTP mode 6 control message format about current state and to request changes in that state. The program runs interactively or uses command line arguments. Requests to read and write arbitrary variables can be assembled, with output options available. `NTPQ` can also obtain and print a list of peers in a common format by sending multiple queries to the server.

The utility uses NTP mode 6 packets to communicate with the NTP server, and hence can be used to query any compatible server on the network which permits it.

**Note:** Since NTP is a UDP protocol, this communication is somewhat unreliable, especially over large distances in terms of network topology. `NTPQ` makes one attempt to retransmit requests, and times out requests if the remote host is not heard from within a suitable timeout time.

## Command Line Format

```
ntpq [ -inp ] [- c command ] [ host ] [ ... ]
```

If command line arguments are omitted, `NTPQ` runs in interactive mode.

### **-c**

The `command` that follows is interpreted as an interactive format command and is added to the list of commands to be executed on the specified host(s). The `command` must be in double quotes if it consists of more than one word. Multiple `-c` options may be given.

### **-i**

Force `NTPQ` to operate in interactive mode. Prompts will be written to the standard output and commands read from the standard input.

### **-n**

Displays all host addresses in dotted quad numeric format rather than converting them to canonical hostnames.

**-p**

Print a list of the peers known to the server as well as a summary of their state. This is equivalent to the `peers` interactive command.

**host**

Sets the host to which future queries are sent, as either a hostname or a numeric address. If `host` is omitted, the local host is used.

## Interactive Commands

Interactive format commands consist of a keyword followed by zero to four arguments. Only enough characters of the full keyword to uniquely identify the command need be typed. The output is sent to the standard output.

## Internal Commands

Internal commands are executed entirely within the `NTPQ` program itself and do not result in NTP mode 6 requests being sent to a server.

**? [command-keyword]**

**help [command-keyword]**

A question mark (?) by itself prints a list of all the known command keywords. A question mark followed by a command keyword prints function and usage information for that command.

**addvars variable\_name [ = value ] [ ... ]**

**rmvars variable\_name [ ... ]**

**clearvars**

The data carried by NTP mode 6 messages consists of a list of items of the form `variable_name = value`, where the `= value` is ignored, and can be omitted, in requests to the server to read variables. `NTPQ` maintains an internal list in which data to be included in control messages can be assembled, and sent using the `readlist` and `writelist` commands described below. The `addvars` command allows variables and their optional values to be added to the list. If more than one variable is to be added, the list should be comma-separated and not contain white space. The `rmvars` command can be used to remove individual variables from the list, while the `clearlist` command removes all variables from the list.

**authenticate yes | no**

Normally NTPQ does not authenticate requests unless they are write requests. The command `authenticate yes` causes NTPQ to send authentication with all requests it makes. Authenticated requests cause some servers to handle requests slightly differently.

**cooked**

Causes output from query commands to be "cooked" for user readability. Variables NTPQ recognizes have their values reformatted for readability. Variables that NTPQ determines should have a decodeable value, but do not, are marked with a trailing question mark (?).

**debug more | less | off**

Turns internal query program debugging on and off.

**delay milliseconds**

Specify a time interval to be added to timestamps included in requests which require authentication. This is used to enable (unreliable) server reconfiguration over long delay network paths or between machines whose clocks are unsynchronized. The server does not now require timestamps in authenticated requests, so this command may be obsolete.

**host [ hostname ]**

Sets the host to which future queries are sent. `hostname` may be either a hostname or a numeric address.

**hostnames [ yes | no ]**

If `yes` is specified, hostnames are printed in information displays. If `no`, numeric addresses are printed instead. The default is `yes`, unless modified using the command line `-n` switch.

**keyid keyid**

This command allows the specification of a 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.

**ntpversion [ 1 | 2 | 3 | 4 ]**

Sets the NTP version number that NTPQ claims in packets. Default is 4. Mode 6 control messages (and modes) didn't exist in NTP version 1.

#### **quit**

Exits NTPQ.

#### **passwd**

This command prompts you to type in a password (which will not be echoed) which will be used to authenticate configuration requests. The password must correspond to the key configured for use by the NTP server for this purpose if such requests are to be successful.

#### **raw**

Causes all output from query commands to be printed as received from the remote server. The only formatting or interpretation done on the data is to transform 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 is twice the timeout value set. If the *milliseconds* value is omitted, the current timeout period is displayed.

## **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 to which the values correspond by including its association ID. An association ID of 0 is special, and 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 printed in some format. Most commands currently implemented send a single message and expect a single response. The current exceptions are the `peers` command, which will send a preprogrammed series of messages to obtain the data it needs, and the `mreadlist` and `mreadvar` commands, which will iterate over a range of associations.

#### **associations**

Obtains and prints a list of association identifiers and peer status for in-spec 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, as described in Appendix B.2.2 of the NTP specification RFC 1305. This is followed by a number of columns containing data decoded from the status word.

```
clockvar [ assocID ] [ variable_name [ = value [ ... ] ] [ ... ]  
cv [ assocID ] [ variable_name [ = value [ ... ] ] [ ... ]
```

Requests that a list of the server's clock variables be sent. Servers which have a radio clock or other external synchronization will respond positively to this. If the association identifier is omitted or zero the request is for the variables of the system clock and will generally get a positive response from all servers with a clock. If the server treats clocks as pseudo-peers, and hence can possibly have more than one clock connected at once, referencing the appropriate peer association ID will show the variables of a particular clock. Omitting the variable list will cause the server to return a default variable display.

### **lassociations**

Obtains and prints a list of association identifiers and peer statuses for all associations for which the server is maintaining state. This command differs from the `associations` command only for servers which retain state for out-of-spec client associations (i.e., fuzzballs). Such associations are normally omitted from the display when the `associations` command is used, but are included in the output of `lassociations`.

### **lpassociations**

Print data for all associations, including out-of-spec client associations, from the internally cached list of associations. This command differs from `passociations` only when dealing with fuzzballs.

### **lpeers**

Like the `peers` command, except a summary of all associations for which the server is maintaining state is printed. This can produce a much longer list of peers from fuzzball servers.

```
mreadlist assocID assocID
```

```
mr1 assocID assocID
```

Like the `readlist` command, except the query is done for each of a range of (non-zero) association IDs. This range is determined from the association list cached by the most recent `associations` command.

```
mreadvar assocID assocID [ variable_name [ = value [ ... ] ] ]  
mrv assocID assocID [ variable_name [ = value [ ... ] ] ]
```

Like the `readvar` command, except the query is done for each of a range of (non-zero) 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 in-spec peers from the internally cached list of associations. This command performs identically to the `associations` except that it displays the internally stored data rather than making a new query.

### **peers**

Obtains a list of in-spec 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 ref ID is unknown); stratum of the remote peer; type of the peer (local, unicast, multicast, or broadcast), when the last packet was received; polling interval (in seconds); reachability register (in octal); and the current estimated delay, offset and dispersion of the peer (all in milliseconds). The character in the left margin indicates the fate of this peer in the clock selection process:

<b>character</b>	<b>rv cmd</b>	<b>Description</b>
<space>	<code>reject</code>	The peer is discarded as unreachable, synchronized to this server (synch loop) or outrageous synchronization distance.
x	<code>falsetick</code>	The peer is discarded by the intersection algorithm as a falseticker.
.	<code>excess</code>	The peer is discarded as not among the first ten peers sorted by synchronization distance and so is probably a poor candidate.
-	<code>outlyer</code>	Discarded by the clustering algorithm

+	<code>candidat</code>	The peer is a survivor and a candidate for the combining algorithm.
#	<code>selected</code>	The peer is a survivor, but not among the first six peers sorted by synchronization distance. If the association is ephemeral, it may be demobilized to conserve resources.
*	<code>peer</code>	The peer has been declared the system peer and lends its variables to the system variables.
o	<code>peer</code>	The peer has been declared the system peer and lends its variables to the system variables. However, a PPS signal is in use

**Note:** Since the `peers` command depends on the ability to parse the values in the responses it gets, it may fail to work with servers that poorly control the data formats.

The contents of the `host` field may be in one of three forms. It may be a hostname, an IP address, or a reference clock implementation name with its parameter. With `hostnames no`, only IP addresses are displayed.

#### **`pstatus assocID`**

Sends a read status request to the server for the given association. (See the `associations` command for `assocIDs`). 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 ]`**

#### **`rl [ assocID ]`**

Requests that the values of the variables in the internal variable list be returned by the server. If the association ID is omitted or is 0 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, which should induce the remote server to return a default display.



```
readvar [ assocID [ variable-name [ = value ] [ ... ] ] ]  
rv [ assocID [ variable-name [ =value ] [ ... ] ] ]
```

Requests that the values of the specified variables be returned by the server, by sending a read variables request. If you omit the association ID or give it as zero, the variables are system variables; otherwise they are peer variables and the values returned are those of the corresponding peer. (See the `associations` command for `assocIDs`). Omitting the variable list sends a request with no data, which should induce the server to return a default display. If more than one variable is requested, separate the variable list with commas and do not include spaces.

```
writevar assocID variable_name [ = value [ ... ]
```

Like the `readvar` request, except the specified variables are written instead of read.

```
writelist [ assocID ]
```

Like the `readlist` request, except the internal list variables are written instead of read.

---

# NTP Management

## Master Server

The MultiNet `Master_Server` process is responsible for starting the NTPD server. When the `Master_Server` process is started, or restarted, it will check the list of enabled servers in the `MULTINET:SERVICES.MASTER_SERVER` file and start those that are enabled. To enable NTP use the following commands:

```
$ multinet configure/servers
MultiNet Server Configuration Utility V5.6
[Reading in configuration from MULTINET:SERVICES.MASTER_SERVER]
SERVER-CONFIG> enable ntp
SERVER-CONFIG> exit
[Writing configuration to
MULTINET_COMMON_ROOT:[MULTINET]SERVICES.MASTER_SERVER]
```

If it is desired that NTP be started immediately, restarting the master server is necessary in addition to the above commands, otherwise NTP will be started the next time MultiNet is started.

## Netcontrol

The MultiNet `NETCONTROL` command is used to start and stop the `NTP_SERVER` process. It can also be used to make certain changes to the operation of NTP, or to inquire about specific items of information.

To start the `NTP_SERVER` process:

```
$ multinet netcontrol ntp
Connected to NETCONTROL server on "LOCALHOST"
< pseudo.process.com Network Control V5.6 at Sun 21-Dec-2019 9:58PM-EST
NTP> start
< Starting NTP server
< NTP server started, process id E9
NTP> quit
```

MultiNet `NETCONTROL` supports the following NTP-specific commands:

DEBUG	Allows setting of the debug level
NOOP	Sends a null command. Useful for making sure the server is running.

NTP-CONTROL-VERSION	Displays the version of the NTP-CONTROL software.
RELOAD	Causes the NTP_SERVER process to restart.
SHOW	Displays some information about current server status and peers.
SHUTDOWN	Causes the NTP_SERVER process to exit.
START	Causes the NTP_SERVER process to be created.
VERSION	Displays the version info for the NTP server and NTP-CONTROL software.
WAYTOOBIG	Allows setting of the panic/waytoobig value

## Monitoring

NTP includes a comprehensive monitoring facility suitable for continuous, long term recording of server and client timekeeping performance. (See the `statistics` configuration option for a listing of each type of statistic currently supported.)

## Troubleshooting Tips

Here are some troubleshooting tips:

- Make sure the entries in the NTP configuration file `MULTINET:NTP.CONF` are correct. At the minimum, there must be a server or peer declaration for a machine that is reachable, and if authentication is enabled, set it up to properly authenticate NTP packets. The machine serving time must be connected either to lower stratum machines or to some reference time source.
- Make sure that the logical `MULTINET_TIMEZONE` is properly defined to reflect the time zone (and daylight savings). If the logical is undefined or incorrect, NTP is likely to abort.
- The `MULTINET_TIMEZONE` logical is defined by MultiNet when the system starts. If MultiNet has already been started, issuing the following command will temporarily redefine the logical:

```
$ MULTINET SET /TIMEZONE=zone name/SELECT=rule name
```

However, the new value will not be preserved through a system reboot. To permanently change the value of this logical, the time zone rules must be configured using `MULTINET CONFIGURE`. However, `MULTINET CONFIGURE` will not redefine the logical on the running system. If MultiNet has already been started, you have to do both.

- If using the `slewalways` command, make sure the system time is within 4000 seconds (or whatever `panic` is set to) of the correct time before starting `NTPD`. If the local system time is off by more than this amount from server time, `NTPD` logs a message and stops running. Also, if the local clock is not within a minute or two of correct time when starting `NTPD` with `slewalways` set, it may take some time for `NTPD` to synchronize the clock. Ideally, set the clock with `NTPDATE` or `SET TIME` before starting `NTPD`.
- Make sure that `TIMED` and `DTSS` services are not running on the system. These services are used to synchronize time, and interfere with `NTP`.
- The following messages are generated by the `NTP` server. They may go to either `OPCOM` or the `MULTINET:NTPD.LOG` file, or both. This log file is the best source of information for troubleshooting in that it contains a record of these messages as well as additional informational messages. Messages appear in the log file without the bracketed prefix. There are four types of messages generated:
  - Configuration messages
  - Peer contact messages
  - Synchronization messages
  - Unexpected error condition messages

Access error messages help by entering

```
$ HELP MULTINET MESSAGES
```

## Troubleshooting Using NTPQ

The `NTPQ` utility has a few commands that are helpful in identifying problems. The `peers` command is one of the simplest and is a quick way to check the offset (time difference) between the local host and peer machines.

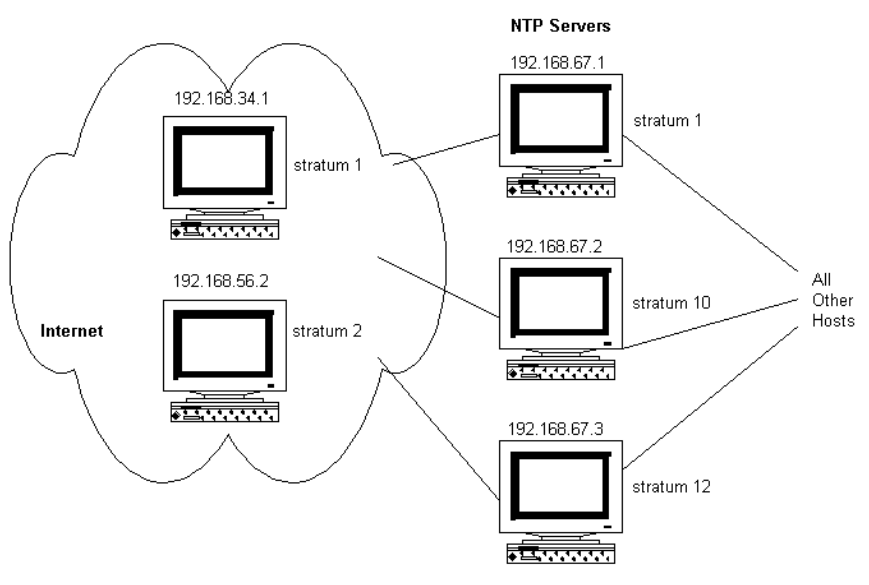
The `readvar` command is useful for more in depth information. Without arguments, it displays information about the local host. When `readvar` is followed by an `assocID`, it displays information about the peer corresponding to the `assocID` (use `associations` to display the `assocIDs` for all peers). Of interest is the record of time offsets and round trip delays for packets (the `filtoffset` and `filtdelay` fields). This provides a record of the last eight time updates obtained from a peer.

The command `readvar assocID flash` displays a useful variable, `flash`, which can be of particular interest for troubleshooting. The bits in the `flash` variable, if set, have the following meaning in relation to a peer:

```
0x01 /* duplicate packet received */
0x02 /* bogus packet received */
0x04 /* protocol unsynchronized */
0x08 /* peer delay/dispersion bounds check */
0x10 /* peer authentication failed */
0x20 /* peer clock unsynchronized */
0x40 /* peer stratum out of bounds */
0x80 /* root delay/dispersion bounds check */
```

## Configuration Example

The diagram below shows a highly redundant and robust configuration with multiple levels of backups. On the Internet close to your network, you have host 192.168.34.1 running at stratum 1, and 192.168.34.2 at stratum 2. In-house, you have host 192.168.67.1 synchronized with a radio clock and configured as a stratum 1 master clock.



As backup servers, you have two hosts, 192.168.67.2 and 192.168.67.3, in the climate-controlled room, one configured at stratum 10 and the other at 12. All other workstations on the floor point to these three servers as their synchronization source. When everything is running, every local host is synchronized to 192.168.67.1, since it is closer than Internet host 192.168.34.1. All the machines (peers) run at stratum 2.

If internal host 192.168.67.1 goes down and the Internet connection is still up, either Internet host 192.168.34.1 or 192.168.34.2 is selected depending on its availability, and the backup servers, 192.168.67.2 and 192.168.67.3, run at stratum 2 or 3, depending on which Internet host was selected. The peers synchronize off 192.168.67.2 or 192.168.67.3 at stratum 3 or 4, again depending on which Internet host was selected.

With 192.168.67.1 still unavailable and the Internet connection lost or all the Internet servers unavailable, 192.168.67.2 runs at stratum 10, since it was configured that way as a local clock. It then becomes the lowest stratum number in the network and all other hosts (including 192.168.67.3) are synchronized to it at stratum 11.

If 192.168.67.2 goes down, 192.168.67.3 runs at stratum 12 and all other hosts synchronize at stratum 13. It is important to set the stratum of 192.168.67.3 to 12. If set to 11, it might have a problem synchronizing to 192.168.67.2, since it may try to synchronize off it but finds it has the same stratum value. 192.168.67.3 would rather synchronize to 192.168.67.2 than to itself.

The below example shows the configuration file entries for each of the three local servers (the other local hosts would all be configured as peers). You do not need to explicitly identify the peer strata, and the order of items is irrelevant.

```
; NTP Configuration on 192.168.67.1
master-clock 1

; NTP Configuration on 192.168.67.2
local-master 10
server 192.168.67.1
server 192.168.34.1
server 192.168.34.2
peer 192.168.67.3

; NTP Configuration on 192.168.67.3
local-master 12
server 192.168.67.1
server 192.168.34.1
server 192.168.34.2
peer 192.168.67.2

; NTP Configuration for Computer Room Host 192.168.67.x
server 192.168.67.1
server 192.168.67.2
server 192.168.67.3
peer 192.168.67.y
peer 192.168.67.z
.
.
.
```



# 12. Configuring Electronic Mail

This chapter describes how to configure the MultiNet SMTP (Simple Mail Transport Protocol) server to send and receive electronic mail.

If you are running PMDF or another mail system that provides its own SMTP support, refer to that mail system's documentation.

## Modifying the MultiNet SMTP Configuration File

The MultiNet SMTP configuration is stored in the `START_SMTP.COM` and `START_SMTP_LOCAL.COM` startup command procedures. Use the `MAIL-CONFIG` utility to edit these files. You start the utility by entering the `MULTINET CONFIGURE /MAIL` command. After using this configuration utility, stop and restart the mail queues. Enter:

- `@MULTINET:START SMTP.COM` to update the VMScluster.
- `@MULTINET:START SMTP_LOCAL.COM` to update the local host only.

## Pipelining and Extended SMTP

The current release of SMTP implements Extended SMTP (RFC-1869) and Pipelining (RFC-2197)



# Delivering Mail to Specific Folders

The SMTP server supports mail delivery to folders other than the `NEWMAIL` folder. The folder names are restricted to UPPERCASE characters only, the pound sign (`#`), and the underscore (`_`). Use of the comma (`,`) in a folder name causes an error. Mail addressed to `user+folder@host` is delivered to the specified folder. You can disable this mechanism by defining the system-wide logical name `MULTINET_SMTP_DISABLE_FOLDER_DELIVERY`.

# Using the Mail Delivery Mechanisms

The current release of SMTP supports alias file extensions that request mail delivery to a file or specify addresses in a separate file. You must use the SMTP aliases file, specified with `MULTINET_CONFIG/MAIL`, to list all of these mail delivery mechanisms. The default is `MULTINET:SMTP_ALIASES`. The syntax for these aliases follows the form of those described in *Configuring Mail Aliases* found later in this chapter. It is necessary to use the colon and semicolon in the command lines as shown in the examples.

<code>&lt;device:[directory]address.list</code>	Delivers mail to the list of addresses in the specified file.  <code>alias1 : "&lt;filespec" ;</code>
<code>  device:[directory]procedure.com parameter(s)</code>	Submits the specified command procedure to the queue identified by the logical name <code>MULTINET_SMTP_BATCH_QUEUE</code> , <code>SYS\$BATCH</code> by default. The first parameter (P1), passed to the submitted procedure, is always the name of a temporary file containing the mail message that the procedure must delete. Any parameter(s) specified in the alias file are passed to the submitted procedure in a single string as its second parameter (P2).  <code>alias2 : " filespec p2 p3" ;</code>

<pre>&gt;device:[directory]mail.file</pre>	<p>Appends mail to the specified file. If no filetype is specified, the default is .yyyy-mm, yyyy and mm are to the current year and month, respectively.</p> <pre>alias3 : "&gt;filespec" ;</pre>
--	--

## Rejecting Mail Messages

The SMTP server supports a set of rules for rejecting mail messages received by itself based on the mail header contents or any combination of MAIL FROM, RCPT TO, and Source IP Address values. Mail matching the criteria can be ignored or rejected quietly with a message to the SMTP client or delivered to an address rewritten according to the rule specification. This capability can be useful for controlling SPAM and preventing your system from being used as a mail relay.

The file `MULTINET:SMTP_SERVER_REJECT.` contains the rejection and rewrite rules. You may specify an alternate file via the logical name `MULTINET_SMTP_SERVER_REJECT_FILE.` A rejection file line of the form `#include device:[directory]reject.file` temporarily suspends processing of the current file and begins processing of the specified file. Rejection files can be nested to arbitrary depth. Comments may be included in rejection files by placing any of the characters `;` or `!` or `#` in the first column of a line. The following is a sample rejection file:

```
!
! This is a sample reject file for the company FLOWERS.COM.
!
! This file is processed sequentially. In other words, processing ends on
! the first rule that the message applies to. So if you have a wildcard
! accept at the top of this file, then no other rules will be processed.
!
! Entries can have one of the following formats:
!
!     from_user [from_ip to_user action action-data]
!
!     :rfc822 header
!
! Wildcards can be used in FROM_USER, FROM_IP, and TO_USER. ACTION is the !
reject action, which is one of:
!
!     n     Don't reject, but rewrite TO address to be ACTION-DATA.
!           If ACTION-DATA is blank then we simply deliver to TO_USER.
!
!     y     Reject and use optional ACTION-FIELD as a rejection message
```

```

!           format that can contain up to three %s formatting
!           designators for mail from, mail to, and local domainname.
!
!           q           Reject quietly -- don't inform Sending SMTP Client that
!           message will be discarded. If only FROM_USER is specified
!           other fields default to FROM_IP=*, TO_USER=*, and ACTION=n.
!
! Don't rewrite or reject any mail to "postmaster*"
!
!           *           * postmaster*           n
!
! Accept all messages with MAIL FROM:<> (bounce messages)
! This rule is commented out because you probably don't want it, although
! We're _supposed_ to always accept it. This is the main method relay
! attacks use, by always saying they are from <> to take advantage of that
! RFC hole.
!
! <>           *           *           n
!
! Reject anything with a Message-ID that appears to have originated from
! cyberpromo.com or nowhere.com
!
! :Message-ID: <*@cyberpromo.com>
! :Message-ID: <*@nowhere.com>
!
! Reject mail from well-known SPAM sites with sample non-standard error
! messages.
!
! <*answerme.com> * * y "Spam from <%s> rejected"
! <*cyberpromo.com> * * y "Spam from <%s> to <%s> rejected"
! <*pleaseread.com> * * y "Spam rejected;%.0s%.0s Contact postmaster@%s"
!
! Disallow percent-hacks (e.g, joe%somewhere.com@flowers.com)
!
! * * *@*@*flowers.com y "No forwarding-path relaying allowed"
!
! Disallow "!" UUCP hacks (e.g. somewhere.com!joe@flowers.com)
!
! * * *!* y "No UUCP relaying allowed"
!
! Rewrite all mail to webmaster to the postmaster
!
! * webmaster*@flowers.com n postmaster@flowers.com
!
!
! Disallow relaying through our mailer, and only allow users on our
! networks to claim to be from our company (flowers.com)
!
! * * *@flowers.com n
! * * *@daisy.flowers.com n
! * * *@[10.0.0.1] n
!
!

```

```

<*flowers.com> 10.0.0.*      * n
<*flowers.com> 10.115.140.*  * n
<*flowers.com> 10.115.141.*  * n
!
! Allow our internal systems to bounce mail out.
!
<>    10.0.0.*      *    n
<>    10.115.140.*  *    n
<>    10.115.141.*  *    n
!
! If a message has slipped through all the tests above, then we want to
! reject it, as they are either relaying through us or it's not a valid
! MAIL FROM.
!
*@*      *      *@*      y      "no relaying through this site"
*        *      *@*      y      "missing domain name in MAIL FROM"
!
!end of sample file

```

Mail rejection rules have two formats:

#### 1. :RFC822\_header pattern

This format causes rejection of any mail in which a line with the specified header matches the given pattern. The following rejection message is sent to the client:

message rejected due to header contents

**Note:** Use caution when rejecting mail based on header contents. No other criteria are considered during rejection processing.

#### 2. from\_user ip\_address to\_user action action\_data

This format causes rejection or alternate delivery of all messages that match all of the patterns specified. The action item can be as follows:

n	Means do not reject the mail, but deliver it to the address specified as the action_data. If action_data is not specified, deliver the message to its intended recipient.
y	Means reject the mail, sending the action_data string to the SMTP client as a rejection message. The action_data item is actually used as a format string and may contain from one to

three %s formatting designators to include the `from_user`, the `to_user`, and the SMTP server name, **in that order**. If `action_data` is missing, the default rejection message is:

```
553-Mail to <to_user> not allowed;  
553 Contact Postmaster@<smtpserver> to remove block
```

q Means reject the mail, but do not give the SMTP client any indication that it has been rejected. Use caution when rejecting messages quietly.

Each of the pattern specifications `pattern`, `from_user`, `ip_address`, and `to_user` may contain the OpenVMS `*` and `%` wildcard characters.

You can represent `from_addr` expressions in the `SMTP_SERVER_REJECT` filter with the `<>` syntax. So, `*@*domain.com` and `<*@*domain.com>` are the same expression. To return to the previous behavior, add the following line to the top of your `SMTP_SERVER_REJECT` file:

```
<> * * n  
(Accept any mail with a MAIL FROM: of <>)
```

When comparing the RCPT TO: address with the `SMTP_SERVER_REJECT` file expressions, any `%` signs in the RCPT TO: address are changed to `@`. You can write filter rules in the `SMTP_SERVER_REJECT` files that can match against forward-path relays. You can add the rule of

```
* * *@*@localdomain y "No forward-path relaying allowed"
```

to your `SMTP_SERVER_REJECT` file above the rules that accept mail with the destination of your domains. RCPT TO: addresses will replace any `%` character with the `@` character for matching purposes only so you can filter with `*@*@*-type` rules. So, RCPT TO: `<xxx%yyy@zzz>` is changed to `xxx@yyy@zzz`. You can use the logical name `MULTINET SMTP_SERVER_REJECT_INFO` to control debug and informational OPCOM messages produced during rejection processing. You should define it to have some non-zero value to request OPCOM messages. The following values may be combined to control message quantity and content:

Values	To show...
1	mail rejected due to action <code>y</code>
2	rewritten addresses (action <code>n</code> with <code>action_data</code> )
4	the reject message sent to the remote system

8	configuration file parsing
16	non-written addresses (action n and no action_data)
32	mail rejected due to action q
64	mail rejected due to header rules

The value 65 is appropriate for auditing rejection activity.

The SMTP service supports both IPv4 and IPv6. There are no configuration parameters to control which one is used, though system managers should be aware of potential reachability issues when switching SMTP to use IPv6.

The remainder of this chapter describes the configuration tasks.

## SMTP Statistics and Accounting

The following sections discuss how to get SMTP statistics and accounting information.

## Network Service Monitoring

Partial support for RFC 2788 (Network Monitoring MIB) has been added to SMTP. To use the 2788 feature, do the following:

```
$ MULTINET CONFIGURE/SERVER
SELECT SMTP
SET FLAGS SNMP_MONITORED
WRITE
EXIT
$ @MULTINET:START_SERVER
```

This feature requires the SNMP Agent X functionality; to use this SNMP must be configured to have Agent X service enabled, and to allow 127.0.0.1 to be an AGENTX\_PEER. See Chapter 23 for more information on SNMP and Agent X.

SMTP's network service monitoring is based on RFC 2788 (Monitoring MIB). Information is maintained only while the service is active. The following items from the Network Services Monitoring MIB (RFC 2788) are available in the enterprises.105.2.25 MIB:

ApplAccumulatedInboundAssociations	(Counter) the total number of connections that the SMTP program has serviced since it was started. enterprises.105.2.25.10
ApplDescription	(String) Description of the program/application. enterprises.105.2.25.16
ApplInboundAssociations	(Counter) The number of connections currently active. enterprises.105.2.25.8
ApplIndex	(Integer) unique application index. The port SMTP is offered on (25). enterprises.105.2.25.1
ApplLastChange	(TimeTicks) the value of sysUpTime when the SMTP program entered the current state. enterprises.105.2.25.7
ApplLastInboundActivity	(TimeTicks) the value of sysUpTime at the time the most recent connection was established. enterprises.105.2.25.12
ApplName	(String) SMTP. enterprises.105.2.25.2
ApplOperStatus	(Integer) the operational status of the SMTP program; the values are: up(1), down(2), halted(3), congested(4), restarting(5), quiescing(6). Some of these values may not be used. enterprises.105.2.25.6
ApplRejectedInboundAssociations	(Counter) the number of connections that have been rejected (due to not being allowed from the access list values). enterprises.105.2.25.14
ApplUptime	(TimeTicks) the value of the SNMP variable sysUpTime when the SMTP program was started. enterprises.105.2.25.5

ApplVersion	<p>(String) the version of the SMTP program. enterprises.105.2.25.4</p> <p>When displaying the enterprises.105.2.25 MIB, entries for 1 to 17 will display but some (specifically .3, .9, .11, .13, .15, .17) will have 0 values.</p>
-------------	--

## Mail Monitoring

Partial support for RFC 2789 (Mail Monitoring MIB) has been added to SMTP. To enable this feature, do the following:

```
$ MULTINET CONFIGURE/MAIL
SET RFC2789 TRUE
WRITE
EXIT
$ @MULTINET:START SMTP
```

This feature requires the SNMP Agent X functionality; to use this SNMP must be configured to have Agent X service enabled, and to allow 127.0.0.1 and the system's own IP address to be an AGENTX\_PEER. See Chapter 23 for more information on SNMP and Agent X.

MtaReceivedMessages	The number of messages received since Message Transfer Agent (MTA) initialization. enterprises.105.3.25.1
MtaStoredMessages	The total number of messages currently stored in the MTA. enterprises.105.3.25.2
MtaTransmittedMessages	The number of messages transmitted since MTA initialization. enterprises.105.3.25.3
MtaReceivedVolume	The total volume of messages received since MTA initialization, measured in kilo-octets. enterprises.105.3.25.4
MtaStoredVolume	The total volume of messages currently stored in the MTA, measured in kilo-octets. enterprises.105.3.25.5



MtaTransmittedVolume	The total volume of messages transmitted since MTA initialization, measured in kilo-octets. enterprises.105.3.25.6
MtaReceivedRecipients	The total number of recipients specified in all messages received since MTA initialization. enterprises.105.3.25.7
MtaStoredRecipients	The total number of recipients specified in all messages currently stored in the MTA. enterprises.105.3.25.8
MtaTransmittedRecipients	The total number of recipients specified in all messages transmitted since MTA initialization. enterprises.105.3.25.9
MtaSuccessfulConvertedMessages	The number of messages that have been successfully converted from one form to another since MTA initialization. enterprises.105.3.25.10
MtaFailedConvertedMessages	The number of messages for which an unsuccessful attempt was made to convert them from one form to another since MTA initialization. enterprises.105.3.25.11

This information can be displayed with the `MULTINET SHOW /SNMP` command. See the `SHOW /SNMP` command in the *MultiNet for OpenVMS Administrator's Reference*.

## Session Accounting

MultiNet can record accounting information from services that have been enabled. Currently this includes FTP and SMTP. The accounting information includes information about when a network session took place and how much data was transferred. The accounting facility is enabled from `MULTINET CONFIGURE/SERVER` and reads `MULTINET:ACCOUNTING.CONF` for additional configuration information. The format of the accounting records is described in `MULTINET_ROOT:[MULTINET.EXAMPLES]ACCOUNTING.H`

A sample program using this is in `MULTINET_ROOT:[MULTINET.EXAMPLES]ACC_DUMP.C`

## Configuring Session Accounting

To configure Session Accounting, follow these steps:

1. Edit the `ACCOUNTING` configuration file, as described in the next section.
2. To start the procedure, do the following:

```
$ MULTINET CONFIGURE/SERVER
ENABLE ACCOUNTING
WRITE
$ @MULTINET:START_SERVER
```

## Configuration File

The Accounting configuration file is `MULTINET:ACCOUNTING.CONF`. The Accounting configuration file defines:

- The Port the Accounting program listens on. This should be an unused port, not the port for the service on which logging is being enabled.
- The name of the file used for accounting records. This file is opened shareable and new records are always appended to it. To start a new file stop the Accounting program, delete (or rename) the existing file, and restart the Accounting program.
- The IP addresses of systems that are allowed to write accounting records to this host.

**Note:** After editing the configuration, stop and restart the Accounting program so that the changes can take effect.

## File Format

Follow these guidelines when entering data in the Accounting configuration file:

- Allow one line for each item.
- Enter information in any order; in uppercase or lowercase.
- Use a pound sign (#) or exclamation point (!) to denote comments. The Accounting facility ignores all information following these characters.

The commands that can be in `MULTINET:ACCOUNTING.CONF` are:

<code>PORT <i>port_number</i></code>	The TCP port that the accounting program should listen on.
<code>PEER <i>ip-address</i></code>	The IP address of a host that is allowed to log records with the accounting software.
<code>FILENAME <i>filename</i></code>	The name of the file that the accounting records will be written to. The <code>MULTINET:device</code> is assumed if a device is not specified as part of the file specification.

## Displaying the Contents of the Logging File

To view accounting information, do the following:

```
$ MULTINET ACCOUNTING/INPUT=accounting_data_file [/output=output_file] -
$ [/since=start_date] [/before=end_date] [/protocol={SMTP, FTP, MAIL}]
[/CSV]
```

- *accounting\_data\_file* is the name of the logging file you want to see.
- *output\_file* is the name of the file you want to call this information. If this field is omitted, the information displays to the terminal screen.
- *start\_date* is the beginning date you want the command to start with. The date format is `[DD-MMM-YYYY [:]] [hh:mm:ss]cc`. If not specified, all records display up to the end of the data found.
  - *DD* is the day of the month, counting from 01.
  - *MMM* is the abbreviation for the month, like JAN, FEB, MAR.
  - *YYYY* is the number of the year, including the century (1999, 2000, 2001, 2002, 2003).
  - *hh* is the hour, from 00 to 23.
  - *mm* is the minute, from 00 to 59.
  - *ss* is the second, from 00 to 59.
  - *cc* is hundredths of seconds.

The time is always in local time.

- *end\_date* is the ending date you want the command to end with. The date format is [DD-*MMM*-*YYYY* [:]] [hh:mm:ss]cc] If not specified, all records display until the end of the file.
- *protocol* is any combination of SMTP, FTP, or MAIL.
- CSV is the Comma Separated Values, for input to products like Excel.

## Accounting File Record Format

The accounting file is written using OpenVMS RMS records. The format of these records is defined in MULTINET\_ROOT:[MULTINET.EXAMPLES]ACCOUNTING.H, and listed below:

```

struct accountingPDU {
    char version;
    char type;          /* type of record */
/*
* FTP:
*   C - Client
*   S - Server
*
* SMTP:
*   N - Network delivery (send)
*   L - Local delivery (received)
*   R - Rejected message
*/
    char flags;        /* not currently used */
    char reserved;     /* for future use */
    int  payload_length; /* length (in bytes) of data after header */
    int  port;         /* IP port of reporting service - 25 SMTP, 21 -
FTP */
    int  reporterIP;   /* IP address of reporter */
};

struct FTPaccounting_data {
    struct accountingPDU header;
    int  start_time[2]; /* VMS time that session started */
    int  end_time[2];  /* VMS time that session ended */
    int  datasent;     /* KBytes of file data sent */
    int  datarecv;     /* KBytes of file data received */
    int  filesent;     /* Number of files sent */
    int  filesrecv;    /* Number of files received */
    int  partnerIP;    /* IP address of partner */
    char user[12];     /* username that operations were done under */
};

struct SMTPaccounting_data {

```

```
struct accountingPDU header;
int  date[2];           /* Time of activity */
int  msg_size;          /* size of message in bytes */
int  from_str_size;     /* size of From: string */
int  to_str_size;       /* size of To: string */
char from_to_str[1];    /* text of From & To string */
};
```

## Configuring SMTP for Accounting

To configure SMTP for accounting purposes, do the following:

```
$ MULTINET CONFIGURE/MAIL
SET ACCOUNTING-HOST hostname
SET ACCOUNTING-PORT port number
WRITE
EXIT
$ @MULTINET:START SMTP
```

The collected accounting information can be displayed with the `MULTINET ACCOUNTING` command.

See the `MULTINET ACCOUNTING` command in the *MultiNet Administrator's Reference*.

## Configuring Mail Parameters

The parameters that control the operations of the MultiNet mailer are described in the table below.

### Configuring Mail Parameters with MAIL-CONFIG

To configure mail parameters with the `MAIL-CONFIG` utility:

1. Start `MAIL-CONFIG` with the `MULTINET CONFIGURE /MAIL` command.
2. Use the `SET parameter_name` commands (for detailed descriptions of these commands, refer to the *MultiNet Administrator's Reference*).
3. Save the configuration with the `SAVE` command.
4. Quit `MAIL-CONFIG` with the `QUIT` command.

The modified configuration takes effect the next time your system reboots or the queues are restarted.

# Mail Parameters

The table below describes all the mail parameters you can set with the MAIL-CONFIG utility.

Parameter	Description
ALIAS-FILE	File in which SMTP aliases are stored; see the <i>Configuring Mail Aliases</i> section.
DECNET-DOMAIN	Domain name for DECnet gateway function; see the <i>Configuring the SMTP-DECnet Mail Gateway</i> section.
DELIVERY-RECEIPTS	Specifies whether mail receipts are sent when incoming mail containing Delivery-Receipt-To: or Return-Receipt-To: headers is submitted to the SMTP queue. If TRUE, mail receipts are sent.
DISALLOW-USER-REPLY-TO	When TRUE, prevents VMS MAIL users from setting a Reply-To: header address with the logical name MULTINET_SMTP_REPLY_TO.
FORWARDER	Identifies a host (known as a <i>mail hub</i> ) to which mail should be forwarded if a host name cannot be resolved for an address. The specified name must resolve to an IP address; it must not resolve to an MX record.
FORWARD-LOCAL-MAIL	Forwards all mail designated for local users to the mail hub instead of delivering it locally. Can be overridden by entries in the SMTP_ALIASES file.
FORWARD-REMOTE-MAIL	Forwards all SMTP-delivered mail to the mail hub instead of directly to the destination host. Can be overridden by a GATEWAY or LOCAL-DOMAIN entry.
HEADER-CONTROL	Controls which RFC-822 headers appear in messages delivered to VMS MAIL users.

HOST-ALIAS-FILE	Contains a list of host names considered aliases for the local host name.
LOCAL-MAIL-FORWARDER	Identifies a host to which mail should be forwarded when a local mail delivery fails because the user name is unknown.
POSTMASTER	Identifies the user name of the system postmaster.
QUEUE-COUNT	Specifies the number of mail processing queues to create on a particular system.
REPLY-CONTROL	Specifies how Internet mail headers should be mapped to the VMS MAIL from header. Permitted values are FROM and REPLY-TO. You may specify both as a comma-separated list.
RESENT-HEADERS	When FALSE, the SMTP symbiont omits the Resent-From, Resent-To, and Resent-Date headers that are usually included when a message is forwarded using a VMS MAIL forwarding address.
RETRY-INTERVAL	Specifies the amount of time (in minutes) that should elapse after a failed attempt before another attempt is made.
RETURN-INTERVAL	Specifies the amount of time (in hours) a message can remain in the processing queue before it is returned to sender.
SEND-BROADCAST-CLASS	Controls the OpenVMS broadcast class used by SMTP for SEND-type messages (which are sent to a terminal).
SMTP-HOST-NAMES	A list of up to 16 host names to consider as aliases for the local host. See the <i>Specifying SMTP Host Aliases</i> section.
START-QUEUE-MANAGER	Determines whether START_SMTP.COM starts the VMS queue manager if it is not already running.

To configure mail parameters via logical name, see the *MultiNet Administrator's Reference*.

# SMTP Configuration Using Logicals

When using a logical name to configure mail parameters, if the setting is for all users, define the logical system-wide. For example:

```
$ DEFINE/SYSTEM/EXECUTIVE MULTINET_SPOOL [PATH.DIRECTORY]
```

## SMTP SYMBIONT LOGICAL

An SMTP SYMBIONT logical may be set up to enable additional logging for debugging purposes.

For example:

```
$ DEFINE/SYSTEM/EXECUTIVE MULTINET SMTP_SYMBIONT_LOG TRUE
```

will enable logging. The default logging filename and location is `MULTINET:MULTINET SMTP_LOG.queueName`. If you wish to change the location and/or filename, you may define:

```
$ DEFINE/SYSTEM/EXECUTIVE MULTINET SMTP_LOG "filespec"
```

where `filespec` is similar to `DEVICE:[dir]filename`. If the filename does not specify an extension, then the `queueName` will be utilized.

## MIME processing

If the logical `MULTINET SMTP_ALLOW_MIME_SEND` is defined to Yes, 1 or True, then if the first line of the message file being sent begins with the mime tag, the blank line at the end of the header section will be suppressed so that the header lines in the mime message file will be seen as header lines rather than message body. The string that is used as the mime tag can be controlled with the logical `MULTINET SMTP_MIME_TAG` which defaults to "Mime-version:"

## Mail Outbound Sanity Checking

Outbound mail sanity checking can be used to test the operation type of the mail message before it is sent out. If there is no operation type, the file is not sent. Use this logical to disable sanity checking:



```
$ DEFINE/SYSTEM/EXECUTIVE MULTINET SMTP_DISABLE_OPTYPE_SANITY_CHECK
```

# Configuring the SMTP Server for Inbound Mail

The MultiNet SMTP server accepts mail from remote hosts and delivers it to users' mailboxes.

By default, the SMTP server is disabled when you install MultiNet. For details on configuring and controlling MultiNet servers, see Chapter 12.

## Translating UNIX-Style Linefeeds to SMTP-Compliant End-of-Line Character Sequences

MultiNet provides a logical name to solve the problem of systems sending messages containing lines terminated by an LF character only instead of the proper CR/LF sequence. The following command tells MultiNet to accept the bare LF as the end-of-line indicator:

```
$ DEFINE/SYSTEM/EXEC MULTINET SMTP_ACCEPT_UNIX_LF TRUE
```

MultiNet lets you validate the contents of the envelope-from field by defining the system-wide logical name `MULTINET SMTP_REJECT_INVALID_DOMAINS`. Use the equivalence string `STRICT` to require the presence of a host in those addresses. For example, require `MAIL FROM: user@host` rather than `MAIL FROM: user`. The host specified in the `MAIL FROM:` address must exist in the DNS database.

The logical name `MULTINET SMTP_ACCEPT_UNIX_LF` has been added as a synonym for `MULTINET SMTP_ACCEPT_UNIX_LF_BRAIN_DAMAGE`. You can define either to have the same effect.

# Configuring the SMTP Server to Limit System/Vendor Information

MultiNet provides you with a way to limit the system/vendor information given out on connection, HELP, and QUIT. The `MULTINET_SMTP_SUPPRESS_VENDOR` logical removes operating system and TCP stack information from SMTP server text responses.

# Configuring the SMTP Symbiont and Mail Queues for Outbound Mail

MultiNet lets users send mail to remote destinations by submitting outbound messages to mail queues that are processed by the MultiNet SMTP symbiont. You can configure the SMTP symbiont to:

- Control users' ability to specify their own REPLY-To: headers (see the *Specifying the REPLY\_TO Header* section.)
- Provide more than one server queue for each cluster node. By default, MultiNet provides one server queue for each cluster node running MultiNet (see the *Configuring Mail Queues* section).
- Forward mail through a central mail hub (see the *Forwarding Mail through a Mail Hub* section.)
- Use gateways to reach specific hosts, domains, or "virtual" domains (see the *Configuring Mail Gateways* section.)
- Use host aliases (see the *Specifying SMTP Host Aliases* section.)
- Use mail aliases (see the *Configuring Mail Aliases* section.)

You can also write your own SMTP dispatcher by modifying and compiling the SMTP user exit `MULTINET_ROOT:[MULTINET.EXAMPLES]USER_SMTP_DISPATCH.C`. Instructions for modifying the dispatcher are outside the scope of this manual.

For outbound mail, MultiNet SMTP eases the 255 character limitation on RFC-822 To: and CC: header lengths. The limit of 255 characters was imposed because some mail applications cannot handle headers longer than 255 characters.

The default header length is 1024 characters. The logical name `MULTINET_SMTP_MAXIMUM_822_TO_LENGTH` can be used to override the 1024 byte default length of the To: and Cc: header fields. The logical can set the maximum length to anywhere from 256 to 65535. To automatically lower the case of usernames in outbound messages, define the logical name `MULTINET_VMSMAIL_LOCASE_USERNAME`.

## Specifying the REPLY\_TO Header

The `MULTINET SMTP REPLY_TO` logical name lets you specify the value for the RFC822 REPLY-TO: header. For example, to set your Reply-To: header to `FNORD@EXAMPLE.COM`, use the command:

```
$ DEFINE MULTINET SMTP REPLY_TO "FNORD@EXAMPLE.COM"
```

This logical name only affects mail agents that use the SMTP% interface (for example, OpenVMS and DECwindows mail). The system manager can disable the use of this logical name with the `SET DISALLOW-USER-REPLY-TO` command of the `MULTINET CONFIGURE /MAIL` utility.

## Disabling VRFY and EXPN

To disable VRFY and EXPN processing, use the logical name `MULTINET SMTP_SERVER_DISABLE_VRFYEXPN`. Define it to have some non-zero value to disable the requisite functions. The following values may be combined to specify which function:

Value	Function
1	to disable VRFY
2	to disable EXPN
3	to disable both VRFY and EXPN

## Configuring Mail Queues

MultiNet uses OpenVMS server queues for SMTP processing. Initially, MultiNet configures each cluster node running MultiNet with a server queue and configures a generic queue for the entire cluster. New messages are placed in the generic queue for processing, which distributes mail processing to the first available server queue.

For example, if three clustered nodes, Huey, Louey, and Dewey, are running MultiNet, MultiNet creates three server queues and one generic queue. The queue names are:

```
SMTP_HUEY           [Execution queue]
SMTP_LOUEY          [Execution queue]
SMTP_DEWEY          [Execution queue]
MULTINET_SMTP       [Generic queue]
```

The following example lists the queues for node Huey:

```
$ SHOW QUEUE MULTINET SMTP/FULL
```

```
Generic server queue MULTINET_SMTP
  /GENERIC=(SMTP_HUEY,SMTP_LOUEY,SMTP_DEWEY) /OWNER=[SYSTEM]
  /PROTECTION=(codes)
```

```
$ SHOW QUEUE SMTP HUEY/FULL
```

```
Server queue SMTP_HUEY, idle, on HUEY::, mounted form DEFAULT
  /BASE_PRIORITY=4 /DEFAULT=(FEED,FORM=DEFAULT) /OWNER=[SYSTEM]
  /PROCESSOR=MULTINET_SMTP_SYMBIONT /PROTECTION=(codes)
```

The queues SMTP\_LOUEY and SMTP\_DEWEY are also created, and are similar to the SMTP\_HUEY queue shown.

**Note:** A standalone (non-clustered machine) has two queues created by default; that is, one generic queue (MULTINET\_SMTP) and one execution queue (SMTP\_nodename).

## Configuring Multiple Queues

If mail traffic is heavy on your system, you can configure multiple server queues on one or more nodes using MAIL-CONFIG. To configure multiple queues with the MAIL-CONFIG utility:

1. Start MAIL-CONFIG with the MULTINET CONFIGURE /MAIL command.
2. Use the SET QUEUE-COUNT command to specify the number of queues on the node (for a full description of this command, refer to the *MultiNet Administrator's Reference*).
3. Save the configuration with the SAVE command.
4. Quit MAIL-CONFIG with the QUIT command.

The modified configuration takes effect the next time your system reboots.

# Configuring Queue Groups

In heterogeneous cluster environments, you may need to partition mail processing by grouping homogeneous subsets of your cluster into queue groups using MAIL-CONFIG. To configure queue groups with the MAIL-CONFIG utility:

1. Start MAIL-CONFIG with the `MULTINET CONFIGURE /MAIL` command.
2. Use the `ADD QUEUE-GROUP` and `DELETE QUEUE-GROUP` commands to add or delete queues (for descriptions of these commands, refer to the *MultiNet Administrator's Reference*).
3. Save the configuration with the `SAVE` command.
4. Quit MAIL-CONFIG with the `QUIT` command.

The modified configuration takes effect the next time your system reboots.

# Forwarding Mail through a Mail Hub

Many sites provide outbound e-mail access to the Internet through a single system known as a *mail hub* to deliver all outbound mail on behalf of the other hosts at the site. A mail hub typically implements a single-address scheme for e-mail users at the site, so that all users have addresses of the form *username@sitename* rather than *username@hostname.sitename*. Site administrators often configure mail hubs to provide Internet e-mail access to hosts that do not have direct access to the Internet. To forward mail through a mail hub:

1. Specify the host that will serve as a mail hub.
2. Specify the conditions under which MultiNet forwards mail to the mail hub.

The following sections describe these procedures.

# Specifying a Mail Hub

To specify the host that will serve as a mail hub for your MultiNet host:

1. Start MAIL-CONFIG (`MULTINET CONFIGURE /MAIL`).
2. Modify the `FORWARDER` parameter:
  - a. With MAIL-CONFIG, use the `SET FORWARDER mailhub_hostname` command.
3. If desired, set any of the following conditions for forwarding mail to the mail hub:
  - a. Forward mail addressed to users on remote hosts (see the *Forwarding Mail Addressed to Remote Hosts* section.)

- b. Exclude hosts in specific domains from remote mail hub forwarding (see the *Excluding Hosts in Specific Domains From Mail Forwarding* section.)
  - c. Forward mail addressed to users on the local host (see the *Forwarding Local Mail* section.)
  - d. Exclude specific local users from mail hub forwarding (see the *Excluding Specific Local Users from Mail Forwarding* section.)
4. Exit the configuration utility. When prompted, save the new parameters.
  5. To make the changes take effect immediately, stop and restart the mail queues. To update the VMScluster, use the `@MULTINET:START_SMTP.COM` command. To update the local host only, use the `@MULTINET:START_SMTP_LOCAL.COM` command. Otherwise, your changes take effect the next time you reboot your system.

## Forwarding Mail Addressed to Remote Hosts

To configure MultiNet to forward mail addressed to remote users via a mail hub:

1. Make sure the `FORWARDER` parameter specifies the host you want to use as a mail hub (see the *Specifying a Mail Hub* section.).
2. Start `MAIL-CONFIG (MULTINET CONFIGURE /MAIL)`.
3. Modify the `FORWARD-REMOTE-MAIL` parameter:
  - a. With `MAIL-CONFIG`, use the `SET FORWARD-REMOTE-MAIL TRUE` command.
4. If desired, exclude hosts in specific domains from mail hub forwarding (see the *Excluding Hosts in Specific Domains From Mail Forwarding* section.).
5. If desired, specify other conditions under which MultiNet forwards mail to the mail hub (see the *Specifying a Mail Hub* section.).
6. Exit the configuration utility. When prompted, save the new parameters.
7. To make the changes take effect immediately, stop and restart the mail queues with `@MULTINET:START_SMTP.COM` to update the VMScluster or with `@MULTINET:START_SMTP_LOCAL.COM` to update the local host only. Otherwise, your changes take effect the next time you reboot your system.

# Excluding Hosts in Specific Domains From Mail Forwarding

If you configure MultiNet to forward mail addressed to remote users via a mail hub (see the *Forwarding Local Mail* section), you can exclude hosts in specific domains from the mail forwarding system by adding the domain to a list of "local domains." To modify the local domain list:

1. Make sure remote mail forwarding is enabled (see the *Forwarding Mail Addressed to Remote Hosts* section.).
2. Start MAIL-CONFIG (MULTINET CONFIGURE /MAIL).
3. To add a domain to the list:
  - a. With MAIL-CONFIG, use the ADD LOCAL-DOMAIN domain\_name command. If domain\_name begins with a dot, it specifies a domain name. Otherwise, domain\_name specifies a host name.
4. To delete a domain from the list:
  - a. With MAIL-CONFIG, use the DELETE LOCAL-DOMAIN domain\_name command.
5. Exit the configuration utility. When prompted, save the modified configuration.
6. To make the new configuration take effect immediately, stop and restart the mail queues with @MULTINET:START\_SMTPL.COM to update the VMScluster or with @MULTINET:START\_SMTPL\_LOCAL.COM to update the local host only. Otherwise, your changes take effect the next time you reboot your system.

## Forwarding Local Mail

To configure MultiNet to forward mail addressed to local users via a mail hub:

1. Make sure the FORWARDER parameter specifies the host you want to use as a mail hub (see the *Specifying a Mail Hub* section.).
2. Start MAIL-CONFIG (MULTINET CONFIGURE /MAIL).
3. Modify the FORWARD-LOCAL-MAIL parameter:
  - a. With MAIL-CONFIG, use the SET FORWARD-LOCAL-MAIL TRUE command.
4. If desired, exclude specific local users from mail hub forwarding (see the *Excluding Specific Local Users from Mail Forwarding* section.).
5. Exit the configuration utility. When prompted, save the new parameters.
6. To make the changes take effect immediately, stop and restart the mail queues with @MULTINET:START\_SMTPL.COM to update the VMScluster or with @MULTINET:START\_SMTPL\_LOCAL.COM to update the local host only. Otherwise, your changes take effect the next time you restart your system.

The logical name `MULTINET_SMTP_APPEND_FORWARDER_TO_MX` can be used to prevent SMTP from appending the forwarder to the MX list by default. To do this:

```
$ DEFINE/SYSTEM/EXECUTIVE MULTINET_SMTP_APPEND_FORWARDER_TO_MX FALSE
```

If the logical name is not defined (or is defined to anything not beginning with F, N, or O), then the FORWARDER is appended to the MX list.

When the logical name `MULTINET_SMTP_IGNORE_INTERFACE_NAMES` is defined (as `/system`, or any value), the MultiNet SMTP mail delivery procedure does not compare the destination address with the addresses of the interfaces on the system to determine if the message could be delivered locally. The default (no logical defined) is to check the addresses of the interfaces on the system. Defining this logical causes the MX records to be used exclusively in determining where a mail message should be delivered.

## Excluding Specific Local Users from Mail Forwarding

If you configure MultiNet to forward local mail via a mail hub (see the *Forwarding Local Mail* section), you can exclude specific local users from the mail forwarding system by creating mail aliases for them in the `MULTINET:SMTP_ALIASES` file. Each users' alias must be in the following format:

`username: *`; For more information on configuring mail aliases, see the *Configuring Mail Aliases* section.

## Configuring Mail Gateways

You can configure MultiNet with gateways to particular hosts or domains to override the normal host lookup used by SMTP or to configure "virtual" domains not actually present on the network. To configure mail gateways with MAIL-CONFIG:

1. Start MAIL-CONFIG with the `MULTINET CONFIGURE /MAIL` command.
2. Use the `ADD GATEWAY` and `DELETE GATEWAY` commands (for descriptions of these commands, refer to *MultiNet Administrator's Reference*).
3. Save the configuration with the `SAVE` command.
4. Quit MAIL-CONFIG with the `QUIT` command.

The modified configuration takes effect the next time your system reboots.



For example, to make it easier for users to address mail to BITNET users, configure a gateway for the .BITNET "domain" to point to one of the Internet-BITNET gateway systems, such as CUNYVM.CUNY.EDU:

```
MAIL-CONFIG>ADD GATEWAY .BITNET CUNYVM.CUNY.EDU
```

Once defined, any mail bound for an address ending in .BITNET is sent to the gateway you specified (in this case, CUNYVM.CUNY.EDU).

## Specifying SMTP Host Aliases

If your system is a member of a VMScluster, you can define *host aliases*, which are host names interpreted by the mailer as aliases for the actual local host name. You can specify these aliases in return addresses for individual users.

## Setting Host Aliases

MultiNet relies on two parameters to obtain its list of host aliases:

SMTP-HOST-NAMES	Is a comma-separated list of up to 16 host aliases. If defined, the first alias in the list is the name used for outgoing mail. Any aliases are names for which your host accepts incoming mail.
HOST-ALIAS-FILE	Is the complete file specification of a file containing an unlimited list of host alias entries (one entry per line). The HOST-ALIAS-FILE value defaults to MULTINET:SMTP_HOST_ALIASES.

To change your host aliases with MAIL-CONFIG, use the SET SMTP-HOST-NAMES command or the SET HOST-ALIAS-FILE command and save the modified configuration with the SAVE command. The new configuration takes effect the next time you reboot the system or the queues are restarted. Alternatively, specify the alias file name dynamically with the following command:

```
$ DEFINE/SYSTEM/EXEC MULTINET SMTP_HOST_ALIASES FILE file-spec
```

If this logical name is not defined, by default the SMTP software looks for the file MULTINET:SMTP\_HOST\_ALIASES. Names in the host aliases file should be listed one per line.

# Specifying Host Aliases for Individual Users

The logical name `MULTINET_SMTP_FROM_HOST` lets you change the host name that appears in your return address on outgoing mail.

Normally, the host name you choose must be a "local" host name; that is, it must be one of the registered SMTP host name aliases on the system (either from the `SMTP-HOST-NAMES` setting or the `HOST-ALIAS-FILE`). If it is not a known alias, the setting is ignored.

If you define the host name in executive mode, however, `MULTINET_SMTP_FROM_HOST` can be any arbitrary host name. The name is not checked against the SMTP host name.

When the logical `MULTINET_SMTP_ENVELOPE_FROM_HOST` is defined the value is used for the host name instead of the actual host name when sending the `MAIL FROM:` line to the remote server. This is useful if there are multiple independent systems that send mail that you would like to appear to be a single system.

This feature lets users from different administrative entities within an organization have return addresses that reflect the names of those entities. To enable this feature:

1. Set up MX records in DNS so mail is routed to the local host for each separate host name. For information about MX records, see the discussion of zone files in Chapter 10.
2. Set up `SMTP-HOST-NAMES` or the `HOST-ALIAS-FILE` with a list of host names.
3. Define the logical name `MULTINET_SMTP_FROM_HOST` for each user. Base the value for this logical name on some aspect of the department or organization to which the user belongs.

# Configuring Mail Aliases

The MultiNet SMTP system supports system-wide mail aliases, system-wide mailing lists, and per-user mail aliases. The default system-wide alias file is `MULTINET:SMTP_ALIASES`. You can configure this name or specify a list of alias file names.

Per-user mail aliases are kept in the file `SMTP_ALIASES` in each user's login directory. The format for alias entries is: `alias: real_address[,...];`

- `alias` is an alphanumeric string.
- `real_address` is either a local or remote electronic mail address.

You can specify multiple addresses by separating them with commas; the alias definition may span multiple lines, if needed, and must always be terminated with a semicolon (;).

For example, a local user has the user name "JB134A", but wants to receive SMTP mail sent to the address "john". The system manager adds the following line to the alias file:

```
john:   jb134a;
```

You can both forward a mail message and deliver it to a local mailbox by adding the mailbox name, preceded by an underscore, to the `MULTINET:SMTP_ALIASES` file. The following example shows such an alias entry:

```
FNORD:  FNORD@SOMEWHERE.EXAMPLE.COM, _FNORD;
```

The leading underscore on the second address (`_FNORD`), tells the SMTP symbiont to skip any further alias processing.

## Mailing Lists

Mailing lists are a special form of mail alias and are supported only in the system-wide alias files. The format for specifying a mailing list is: `list-name:: owner-address, file-spec;`

- A double-colon (`::`) signifies that this alias is a mailing list.
- `owner-address` is the address of the mailing list owner. Messages sent to this mailing list go to each subscriber on the list with the return-path set to this address. The owner address can be an actual user's address or an alias, if desired.
- `file-spec` is the file specification for the file containing the subscribers to the mailing list. Specify a complete path name for this file, including the device and directory.

For example, you might want to set up a mailing list called `OPERATIONS-STAFF` for your operations staff, and have your operations manager, user `OPER1`, manage that list. You might set up the mailing list this way:

```
Operations-Staff:: Operations-Manager, USERS:[OPER1]STAFF.LIST;  
Operations-Manager: OPER1;
```

Mail sent to `OPERATIONS-STAFF` is forwarded to the addresses listed in `USERS:[OPER1]STAFF.LIST`. Because this file is in `OPER1`'s area, the operations manager has control over who is included in the list. The list is set up in this example so the return-path on list messages is set to "Operations-Manager" instead of user `OPER1`; setting up the list owner as an alias makes it easier to change list owners at a later date.

## Specifying the System-Wide Mail Alias File

By default, the MultiNet SMTP system obtains system-wide mail aliases from the `MULTINET:SMTP_ALIASES` file. You can configure MultiNet to use any other file, or to use multiple files.

To change the SMTP aliases file with `MAIL-CONFIG`, use the `SET ALIASES-FILE` command, then save the modified configuration with the `SAVE` command. The new configuration takes effect the next time you reboot the system.

## Using Mail Aliases and Mailing Lists From VMS MAIL

If you want aliases configured within the MultiNet SMTP alias file to be accessible to local VMS MAIL users (or those connected via DECnet), specify the address using the MultiNet SMTP VMS MAIL foreign mail protocol interface.

For example, a local user wanting to send mail to the "gcc-users" mailing list would specify the address `SMTP%"gcc-users"`.

**Note!** You can, however, define a VMS MAIL alias containing the `SMTP%` specification. To define the VMS MAIL alias "Operations-Staff," use the `VMS MAIL SET FORWARD` command:

```
MAIL> SET FORWARD SMTP%""Operations-Staff-USERS"" /USER=Operations-Staff
MultiNet SMTP uses the RFC-822 To: and CC: headers to provide the contents of the VMSmail To: and CC: fields. To enable this processing, define the logical name
MULTINET_VMSMAIL_USE_RFC822_TO_HEADER.
```

**Note:** VMSmail limits the length of its To: and CC: fields to 255 characters.

## IMAP Server

The Internet Message Access Protocol (IMAP) server lets an IMAP-compliant client mail program access remote message storage as if the storage were local. MultiNet's implementation is based on IMAP Version 4, Revision 1.

IMAP and the Post Office Protocol (POP3), described in the next section, operate differently. IMAP retains the message on the server, whereas POP3 retrieves the message and stores it "off-line" on the client, thereby deleting it from the mail server. IMAP does not delete the mail message and lets you access your mail from more than one client workstation at a time.

IMAP was designed to:

- Be fully compatible with Internet messaging standards, such as MIME.
- Allow message access and management from more than one computer.
- Allow access without relying on less efficient file access protocols.
- Provide support for "online," "offline," and "disconnected" access modes
- Support concurrent access to shared mailboxes.
- Eliminate the need for the client software to know about the server's file storage format.

The IMAP protocol includes operations for:

- Creating, deleting, and renaming mailboxes
- Checking for new messages
- Permanently removing messages
- Setting and clearing flags
- Server-based RFC-822 and MIME parsing and searching
- Selective fetching of message attributes, texts, and portions thereof, for efficiency

For other IMAP features and how they contrast with those of POP3, see either of these web sites:

<http://www.imap.org/imap.vs.pop.brief.html>

<http://www.imap.org/imap.vs.pop.html>

## Inhibiting Output in Command Procedures for the IMAP Service

Problems arise when remote users log into systems using a login command procedure (SYS\$LOGIN:SYLOGIN.COM or SYS\$MANAGER:SYLOGIN.COM) that requires screen output. To inhibit this behavior, make sure the following lines are included at the top of all login command procedures:

```
$ VERIFY = 'F$VERIFY(0)           ! Turn off verify without echoing
$ IF F$MODE() .EQS. "OTHER" THEN EXIT ! If a DETACHED process (IMAP)...
$ IF VERIFY THEN SET VERIFY       ! If a batch job, may want to turn
                                   ! verify back on.
```

# IMAP Mail Folders

In contrast to POP3, IMAP allows you to access server mail folders (message stores) other than INBOX. In MAIL, for example, if you create a NOTES folder, you can access mail in that folder. This NOTES folder can be in a mail file other than the default MAIL.MAI file. In fact, you can set a configuration parameter that determines the way mail folders are presented to the client so that you can use folders in these other mail files.

Your default mail directory includes a .IMAPRC file in which you can set certain configuration directives (described more fully in the *Error! Reference source not found.* section that follows). Among these directives is `allow-subfolders`. This directive specifies that folder names are comprised of a directory (optional), mail file, and folder. For example, the NOTES folder in MAIL.MAI is represented as `mail/notes` (as opposed to just `notes` if the directive were not set). This would distinguish it from another NOTES folder in the OLD.MAI mail file, for example, which would be named `old/notes`.

Each level beyond the second in this hierarchy represents a subdirectory of the default mail directory. For example, the NOTES folder in `[.ARCHIVED]MAIL.MAI` has the IMAP equivalent of `archived/mail/notes`.

Because of this folder syntax ambiguity, directory names, file names, and folders can overlap, such as the examples below.

This mail file...	Containing this folder...	Has this IMAP equivalent...
MAIL.MAI	NOTES	mail/notes
[.MAIL]NOTES.MAI	STUFF	mail/notes/stuff
[.MAIL.NOTES]STUFF.MAI	BOBS	mail/notes/stuff/bobs

Entries in the syntax can at different times be mail files, directories, subdirectories, or folders. Because of this overlap, the server must keep an internal representation of the hierarchy and mark what each level of the folder name means. This information is critical when renaming or deleting folders.

One restriction is that a first level folder (MAIL, for example) cannot be a message store, since it represents only a file and not a mail folder. INBOX, however, is a special case. INBOX is always INBOX, cannot be deleted or renamed (a rename moves messages to the renamed folder but does not delete INBOX), and never goes away. In IMAP, INBOX is `mail/NEWMAIL` by default, and is hidden to the user.

(Note that you can change the mail "in-box" from INBOX to another folder by defining the `file-inbox-messages-to-folder` directive in the `.IMAPRC` file. See the next section for details.)

You can also access mail files in your login directory (`SYS$LOGIN`) by prefixing the folder name with a tilde (`~`). The `~` folder is reserved and cannot be used by other folders.

## IMAP Directives File

Users can set certain preferences by creating a file in their default mail directory called `.IMAPRC` and including directives. The below table lists these directives along with their meanings. Each directive must be on its own line and in lowercase.

This directive...	Does the following...
<code>set allow-child-folders</code>	Enables or disables the use of subfolders and the way that folders are presented to the client. By default, this value is <code>false</code> . If you want to set the value to be true, put this line in the <code>.IMAPRC</code> file: <code>set allow-child-folders true</code> . (See the previous section for details.)
<code>set autofile-messages-to-folder foldername</code>	Moves read messages from INBOX to the specified folder in the user's default mail file. By default, this option is disabled.
<code>set case-insensitive-folders</code>	Specifies that folder names are case-insensitive. Otherwise, two folders with the same name but with different cases could become inaccessible to the IMAP client. Newly created folders are created in uppercase on the server. By default, this value is <code>false</code> . If you want to set the value to be true, put this line in the <code>.IMAPRC</code> file:  <code>set case-insensitive-folders true</code>
<code>set do-purge-reclaim</code>	Enables or disables purge-reclaim operations upon closing a folder. By default, this value is <code>true</code> .

<code>set folder-timer <i>delta_time</i></code>	Specifies the frequency the IMAP server will check for externally created folders. By default, this value is 00:02:00 (2 minutes).
<code>set inbox-folder <i>foldername</i></code>	Maps INBOX to the specified folder in the user's default mail file. By default, this value is NEWMAIL.
<code>set newmail-timer <i>delta_time</i></code>	Specifies the frequency the IMAP server will check folders for new messages. (Note that checking for new mail is time consuming for large folders.) By default, this value is 00:00:30 (30 seconds).

## IMAP Options in the Global IMAPD.CONF file

These options are valid in the global IMAPD.CONF file (MULTINET:IMAPD.CONF) as shown in the below table:

This directive...	Does the following...
<code>set decnet-address nodename namespace domainname</code>	Maps the specified decnet nodename and/or namespace to a given domain name. Use " " to represent either a blank nodename or namespace. Multiple entries are allowed. By default, this option is disabled. Example:  <code>set decnet-address knob " " door.com.</code>
<code>set enable-full-cache</code>	Enables or disables full message caching. By default, this value is false.
<code>set max-ping-count <i>integer</i></code>	Specifies, in number of messages, the threshold at which the server will no longer attempt to check for new messages. By default, this value is 20000.



<pre>set smtp-transport-prefix string</pre>	<p>Specifies the SMTP transport prefix. By default, this value is SMTP. If you want to change it to MX, put this line in the IMAPD.CONF:</p> <pre>set smtp-transport-prefix MX</pre>
<pre>set trailing-header-marker string</pre>	<p>Specifies a text string used to indicate the start of RFC822 message headers if the system does not place them at the start of the message. By default, this option is disabled.</p>

## IMAP State Information Files

The IMAP server includes files created in the user's mail directory where it maintains state information, as shown in the following table.

This file...	Stores...
.MAILBOXLIST	Folders to which the user subscribes.
.NEWMAILBOXES	List of folders known to be empty. OpenVMS MAIL deletes folders once it deletes the last message, so that the server must "remember" these folders.
mailfolder.foldernameuidvalidity	For each folder, the UIDs for all the messages. The file name is composed of the folder name and its UIDVALIDITY code. For example: MAIL.NEWMAIL3B3200E6. In the example, the folder name is NEWMAIL and the UID validity code is 3B3200E6.

## IMAP Logicals

The following IMAP logicals are supported:

## MULTINET\_IMAPD\_MESSAGE\_ONE

By default, an informing message of `This message cannot be retrieved` is sent to the user when the processing message is too big. You can use this logical to define a different informing message. For example, if you define the logical like this:

```
$ define/sys/exe MULTINET_IMAPD_MESSAGE_ONE "Your mail is too big to be processed"
```

The message seen by the user is:

```
Your mail is too big to be processed.
```

If the defined logical string starts with `@`, the rest of the string defines a text file that holds the text. The maximum file size of the informing message is 255 bytes. For example, if you define the logical like this:

```
$ define/sys/exe MULTINET_IMAPD_MESSAGE_ONE "@MULTINET:IMAPD_MSG_ONE.TXT"
```

and edit the file `MULTINET:IMAPD_MSG_ONE.TXT` to be:

```
The size of this message is too big for the IMAP server to process. Most likely it has a big attachment file. Contact the sender to arrange re-transmission by other means such as FTP.
```

```
-System Manager.
```

The text message from the System Manager is seen by the user when the processing message cannot be retrieved.

## MULTINET\_IMAPD\_MESSAGE\_SIZE\_LIMIT

There could be times when an oversized mail file prevents the mail system from working. This oversized mail file could also slow down other processes on the system. If such a case happens, use this logical to limit the size of the mail that IMAP processes.

When a mail file size reaches the limit defined by this logical, IMAP does not process it and sends a message to the user. Use the following metrics to define this logical: "S" (40K records), "M" (120K records), or "L" (240K records).

**Note!** Because VMS uses records to define the mail size and a record could have as few as 1 byte to as many as 255 bytes, the size limit defined by the logical does not reflect the actual mail size in bytes. For example:

```
$ define/sys/exe MULTINET_IMAPD_MESSAGE_SIZE_LIMIT "M"
```

A mail with a 5MB attachment file could reach the limit. But another mail with a 6MB attachment file could pass the limit.

**Note:** If the logical is not defined, the mail size limit is actually the current VMS Page file limit quota (PGFLQUO) of the IMAP process.

## **MULTINET\_IMAPD\_LOGLEVEL *n***

This logical enables additional logging for debugging purposes. Output is written to the file `MULTINET:IMAP_SERVER.LOG`. By default, this logical is unassigned. IMAP events normally are logged to `SYSLOG`.

## **MULTINET\_IMAP\_UPDATE\_LOGIN\_TIME**

This logical updates the last non-interactive login field in this SYSUAF for each IMAP login.

# **Post Office Protocol (POP) Versions 2 and 3**

The MultiNet Post Office Protocol version 2 (POP2) and version 3 (POP3) servers allow a remote user to access mail stored in VMS MAIL files from POP2 and POP3 mail clients. The POP2 protocol is documented in RFC-937, "Post Office Protocol: Version 2," and POP3 is documented in RFC-1460, "Post Office Protocol: Version 3."

The remote user must have a valid account on the OpenVMS host and mail must be delivered into the VMS MAIL files associated with the account. A user specifies a user name and password when VMS MAIL is accessed from a POP2 or POP3 client.

The POP3 server accepts Kerberos V4 authentication in place of user name and password authentication. The current release of the Eudora mail client supports this method of authentication. The POP3 server allows cross-realm Kerberos V4 authentication when the logical name `MULTINET_POP3_ENABLE_CROSS_REALM_AUTHENTICATION` is defined.

The POP3 server waits indefinitely for input from POP3 clients. A logical is definable so you can specify the amount of time the POP3 server should wait for input before closing the connection. The logical is `MULTINET_POP3_INPUT_WAIT x`, where `x` is a normal VMS time string.

The POP3 server supports Unique Identifier Lists (via the `UIDL` command) for improved operation of Eudora clients.

The current release of the MultiNet POP3 server does not support the `APOP` authentication mechanism. The user name and password are validated by `LOGINOUT`, and a server process is created on the OpenVMS host. The server process invokes both the system-wide login command procedure (`SYS$MANAGER:SYLOGIN.COM`) and the user's `LOGIN.COM` before the POP server image is run. A log file is created in the user's login directory detailing the transactions between the client and the OpenVMS system; with POP2 this log file is called `POP2_SERVER.LOG` with POP3, it is called `POP3_SERVER.LOG`.

## POP Logical Names

You can define the logical names described in this section to provide additional functionality to POP2 and POP3. These logicals, which affect several aspects of mail operation, can be defined either system-wide or for individual users.

**Note:** The default value for all of these logicals is 0 (all bits disabled).

You can use these logicals for either POP2 or POP3. In using the logicals, substitute either 2 (for POP2) or 3 (for POP3) for the `x` in the logical name.

### **MULTINET\_POP3\_UPDATE\_LOGIN\_TIME**

When defined `/SYSTEM`, this causes the `LAST NONINTERACTIVE DATE/TIME` field in `SYSAUF` to be updated with the time the POP3 session was started. Note that this logical applies to POP3 sessions only.

# Specifying POP Functions Using the MULTINET\_POPx\_FLAGS Logical

This logical specifies functions as decimal values that are interpreted as described in the following table. To define more than one function, add the decimal values together; for an example, see the *Defining the Logicals System-Wide* section.

Value	Effect
1	Read only messages marked as new from the NEWMAIL folder. The default is to read all messages in this folder. If MULTINET_POPx_SOURCE_FOLDER is defined, this flag is ignored.
2	Move messages from the NEWMAIL folder to the MAIL folder after they are read. If MULTINET_POPx_DEST_FOLDER is defined, this flag is ignored.
4	Release the POP client before the VMS mailbox is actually closed; that is, cause a quick close. VMS MAIL can take several minutes to reclaim large amounts of deleted message space. The server replies to the client with a success, closes the connection, releases the client, and closes the mailbox. Any errors that result from closing the mailbox when this flag is enabled are lost.
16	<p>Remove the "NODE:." portion of the "From:." address. The MultiNet POP server creates RFC-822 headers from VMS headers before sending the message to the POP client. If the mail message was received via SMTP, SMTP headers are used.</p> <p>When NODE is the same as the OpenVMS logical name SYS\$NODE, the "NODE:." portion of the address is removed; otherwise, this portion of the address is retained.</p> <p>For example:</p> <ul style="list-style-type: none"><li>• For a local node WHITEFANG in which a message originates via DECnet from a remote OpenVMS host, the "From:." line is WHARFIN::HOLMES.</li><li>• After conversion to RFC-822 conventions, the line is "WHARFIN::HOLMES"@WHITEFANG.EXAMPLE.COM.</li><li>• Because some POP clients cannot reply to this address, the bit value represented by 16 should be enabled. When enabled, the "From:." line becomes HOLMES@EXAMPLE.COM.</li></ul>

	<p><b>Note:</b> If a message is routed via DECnet before being received by SMTP, enabling this bit invalidates the return path to the remote DECnet node.</p>
32	Deleted messages are saved into the folder specified by <code>MULTINET_POPX_DEST_FOLDER</code> . Otherwise, when a message is deleted, it is removed completely.
64	<p>POP builds headers from the OpenVMS Mail "From" statement. Specify this flag when your mailer has been configured to not insert the SMTP headers as the first lines of text in the message.</p> <p>The only time you should use this value in the <code>MULTINET_POP3_FLAGS</code> logical name is when you want to construct ad hoc SMTP headers (that is, when your SMTP agent is configured to place the real headers at the bottom of the message).</p>
128	The mail box is compressed, but all versions of <code>MAIL.OLD</code> in the <code>MAIL</code> directory are deleted.

## Setting the `MULTINET_POPx_DEST_FOLDER` and `MULTINET_POPx_SOURCE_FOLDER` Logicals

The `MULTINET_POPx_DEST_FOLDER` logical specifies the folder containing messages that have been read.

The `MULTINET_POPx_SOURCE_FOLDER` logical specifies the folder from which to read messages. The default folder name is `NEWMAIL`.

By default, the POP2 and POP3 servers look for mail in the `NEWMAIL` folder of the user's OpenVMS `MAIL` files. This default may be overridden by defining the logical names

`MULTINET_POPx_SOURCE_FOLDER` and `MULTINET_POPx_DEST_FOLDER` in the file

`SYSS$LOGIN:LOGIN.COM`. For example, if a POP3 user wants to access mail stored in the `MAIL` folder by default, place the following command in `LOGIN.COM`:

```
$ DEFINE MULTINET POP3 SOURCE FOLDER "MAIL"
```

**Note:** OpenVMS MAIL folder names are case-insensitive.

When a mail message is accessed by POP2 or POP3, it remains in its original folder until the POP client deletes it. This happens even if the mail is being accessed from the user's `NEWMAIL` folder. Users may, however, define the logical name `MULTINET_POPx_FLAGS`, using the value 2, in their `LOGIN.COM` files to alter this behavior. If `MULTINET_POPx_FLAGS` is set to 2, mail messages are placed in a user's `MAIL` folder. This occurs after they are read via POP from the `NEWMAIL` folder, if the POP client does not delete the messages after it reads them.

This behavior is the same as with OpenVMS MAIL. For example, if a POP2 user wants mail read from the `NEWMAIL` folder placed in the `MAIL` folder after being read, add this command to `LOGIN.COM`:

```
$ DEFINE MULTINET POP2_FLAGS "2"
```

Again, the user must also configure the POP2 client so messages are not deleted after they are read. If the client deletes a message, it is not saved in either the `MAIL` or `NEWMAIL` folder on the OpenVMS server.

## Defining the Logicals System-Wide

All POP logical names can be defined system-wide for all users, as shown in the following example:

```
$ DEFINE /SYSTEM /EXECUTIVE MULTINET_POP3_FLAGS "7"
```

This example sets the flags parameter so that:

- All users only read messages from the `NEWMAIL` folder that are marked as new.
- Messages are moved from the `NEWMAIL` folder to the `MAIL` folder after they are read.
- The POP client is released quickly at the end of the mail session.

# Configuring SMTP Service for ALL-IN-1 Users

The MultiNet mailer supports users of HP's ALL-IN-1 office automation environment (often referred to as ALL-IN-1 IOS or ALL-IN-1 Classic) and ALL-IN-1 MAIL via an interface to Message Router, the backbone of HP's MAILbus product line. This interface allows both ALL-IN-1 IOS and ALL-IN-1 MAIL users to send and receive SMTP mail. Message Router V3.1 or later is required for this feature to function properly. For information on sending and receiving SMTP mail from within ALL-IN-1 IOS and ALL-IN-1 MAIL, see the electronic mail chapter in the *MultiNet User's Guide*.

## Before Configuration

You must have Message Router V3.1 or later installed on your system before configuring the MultiNet SMTP to Message Router (SMTP/MR) interface. If you want to support automated conversion of WPS and DX documents in ALL-IN-1 messages to ASCII, you must also have the Message Router VMS MAIL Gateway (MRGATE) installation kit.

You do not need to install MRGATE on your system; however, certain object libraries in the MRGATE kit are needed to provide the necessary document conversion functions. The SMTP/MR gateway software functions even if the document conversion is not built. It does, however, cause all WPS and DX message bodyparts to be discarded as the ALL-IN-1 message passes through SMTP/MR.

## Configuring SMTP/MR

The `MR_CONFIGURE.COM` command procedure in the `MULTINET:` directory is used to configure the SMTP/MR gateway software. Execute this procedure with the DCL command:

```
$ @MULTINET:MR_CONFIGURE
```

The command procedure presents a series of prompts. Enter a question mark (?) at any time to display more information about that prompt. The configuration command procedure prompts you for the following information:

1. Whether to display a detailed explanation before each question. It is recommended that you answer YES to this question the first time you run the configuration procedure.
2. The domain name of the gateway system. This is a domain-style host name used to refer to the gateway from the Internet. Be sure the name you choose is within a domain or subdomain over



which you have administrative authority, and is not currently being used for another host. If your local domain is EXAMPLE.COM, a reasonable choice for this domain name would be MR.EXAMPLE.COM. This name is largely for internal use, and should not be needed to address mail to ALL-IN-1 users.

3. The domain name to be used for local ALL-IN-1 IOS users. This is a domain-style host name used to indicate to the MultiNet mailer that it should pass the message to the Message Router for delivery to an ALL-IN-1 user. Be sure the name you choose is within a domain or subdomain over which you have administrative authority, and is not currently being used for another host. If your local domain is EXAMPLE.COM, a reasonable choice for this domain name would be A1.EXAMPLE.COM. If you are not running ALL-IN-1 IOS, you should specify NONE.
4. The domain name to use for local ALL-IN-1 MAIL users. This is a domain-style host name used to indicate to the MultiNet mailer that it should pass the message to Message Router for delivery to an ALL-IN-1 user. Be sure the name you choose is within a domain or subdomain over which you have administrative authority, and is not currently being used for another host. If your local domain is EXAMPLE.COM, a reasonable choice for this domain name would be AM.EXAMPLE.COM. If you are not running ALL-IN-1 MAIL, you should specify NONE.
5. The Message Router mailbox name used for the gateway. This Message Router mailbox name is used by ALL-IN-1 users to send outbound SMTP mail. You are directed later to create this mailbox with the Message Router MRMAN utility. The default value of "SMTP" should be used.

Both ALL-IN-1 IOS and ALL-IN-1 MAIL users use the address form *user@host@SMTP* to specify remote SMTP recipients. Each ALL-IN-1 mail utility places this outbound mail in the Message Router mailbox named SMTP. The MultiNet mailer "picks up" mail from this mailbox and sends it via the normal SMTP delivery mechanism.

The current version of MultiNet allows both ALL-IN-1 IOS and ALL-IN-1 MAIL users to reply to messages imported with the V command or forwarded into ALL-IN-1 using an address of the form MRGATE:"A1::user" or MRGATE::"AM::user". Return addresses are translated from Message Router format to a more standard RFC-822 format in a fashion analogous to the DECnet to SMTP gateway conversion. The following logical names can be used to customize the translation:

MULTINET SMTP MRGATE_NAME	Change the name of the Message Router gateway mailbox. The default value is MRGATE.
MULTINET SMTP A1_NAME	Change the name of the ALL-IN-1 IOS gateway mailbox. The default value is A1.
MULTINET SMTP AM_NAME	Change the name of the ALL-IN-1 MAIL gateway mailbox. The default value is AM.

MULTINET SMTP_A1_DOMAIN	Specify the RFC-822 domain associated with the ALL-IN-1 IOS gateway. Use the domain you specified when configuring your SMTP/MR gateway.
MULTINET SMTP_AM_DOMAIN	Specify the RFC-822 domain associated with the ALL-IN-1 MAIL gateway. Use the domain you specified when configuring your SMTP/MR gateway.

6. The Message Router mailbox name used for delivery to ALL-IN-1 MAIL users. This Message Router mailbox is used to deliver inbound SMTP mail to ALL-IN-1 MAIL users. Normally this mailbox is named "AM", but the name is configurable. If your site uses a name other than AM, enter that name.
7. The Message Router mailbox name used by default when mail is sent to the domain name of the gateway system (for example, MR.EXAMPLE.COM). Specify the default value, "MRGATE," to indicate the Message Router VMS MAIL Gateway mailbox.
8. The Message Router mailbox name for the local system. Specify the DECnet node name of the local system. SMTP/MR uses this name to generate addresses that are valid on remote systems.
9. The names of any null routes. Specify the name of the local DECnet node and any other nodes in a homogeneous VMScluster (including the cluster alias). Message Router routing information often includes local DECnet node names or a cluster alias which is superfluous in outbound SMTP mail. To determine the names of any null routes, use the MRMAN utility `SHOW *` command. Null routes have the general form:

```
nullname,          Replace=
```

The names you specify at this point are automatically stripped from addresses passing through SMTP/MR, greatly simplifying them when they pass into the SMTP world. Use a blank entry to terminate the list.

10. The SMTP address of the local postmaster, which should be the full domain-style e-mail address of the user who should receive error messages generated by SMTP/MR.
11. The password associated with this gateway's Message Router mailbox (SMTP by default). Message Router protects each mailbox with a password to ensure privacy of mail messages.
12. Whether you want to have messages with WPS and DX bodyparts automatically converted to ASCII. SMTP/MR can perform these conversions if it is linked against libraries provided as part of various Message Router products, notably MRGATE. If you intend to have messages in these special formats converted to ASCII, answer YES. If you answer NO, these bodyparts are not converted, and may be discarded.
13. You are then prompted for the names of several SMTP/MR configuration files. Accept the defaults for each of these file names.

14. You are then asked if you want to generate the configuration files. If you are satisfied with all of your responses, type YES.

In the following example, which shows an MR\_CONFIGURE.COM session, the local host's DECnet node name is MIGUEL, while its TCP/IP host name is MIGUEL.EXAMPLE.COM. The DECnet cluster alias is IRISDN; and other hosts in the homogeneous VAXcluster are named WHARFIN, BIGBTE, and YAYA. The e-mail address of the local postmaster is POSTMASTER@EXAMPLE.COM.

```
$ @MULTINET:MR_CONFIGURE

SMTP-MR Configuration Utility, Version V3.1
This utility creates an initial pair of databases for mapping SMTP RFC 822-
style addresses to Message Router X.400-style addresses and back again. Only
minimal mappings are created to support ALL-IN-1 users sending and receiving
SMTP mail. If you need full MAILbus to SMTP gatewaying capabilities, contact
Process Software at (508) 879-6994 about their PMDF and PMDF-MR e-mail
gateway products.

Important note: No changes are made to existing SMTP-MR database
information until all questions have been answered. This utility can be
aborted at any prompt by entering a CTRL/Z. The files output by this utility
may optionally be redirected to a different location so they will have no
impact on the existing SMTP-MR databases.

Do you wish to continue [Y]? RETURN
Do you wish to have a detailed explanation printed before each question [N]?
RETURN
Domain name of the gateway: MR.EXAMPLE.COM
Domain name for local ALL-IN-1 IOS users [A1.EXAMPLE.COM]: RETURN
Domain name for local ALL-IN-1 MAIL users [AM.EXAMPLE.COM]: RETURN
Message Router mailbox name for the gateway [SMTP]: RETURN
ALL-IN-1 IOS mailbox known to Message Router [A1]: RETURN
ALL-IN-1 MAIL mailbox known to Message Router [AM]: RETURN
Message Router mailbox used by default [MRGATE]: RETURN
Local system's Message Router mailbox [MIGUEL]: RETURN

Local node name or other null routes [RETURN if no more]: MIGUEL
Local node name or other null routes [RETURN if no more]: WHRFIN
Local node name or other null routes [RETURN if no more]: BIGBTE
Local node name or other null routes [RETURN if no more]: YAYA
Local node name or other null routes [RETURN if no more]: IRISDN
Local node name or other null routes [RETURN if no more]:

RFC822 address of local PostMaster: postmaster@example.com
Password for the Message Router mailbox: secret
Convert WPS-PLUS and DX messages to ASCII automatically [Y]? RETURN
SMTP to MR mapping text file [MULTINET:TO_MR.]: RETURN
MR to SMTP mapping text file [MULTINET:FROM_MR.]: RETURN
Gateway options file [MULTINET:MR_OPTIONS.]: RETURN
SMTP-MR checklist file name [MULTINET:SMTP-MR.CHECKLIST]: RETURN
```

```

All configuration questions have been answered.
Do you wish to generate the configuration files [Y]? RETURN
Generating SMTP to MR mapping text file...
SMTP to MR mapping text file is complete.

Generating MR to SMTP mapping text file...
MR to SMTP mapping text file is complete.

Generating the setup checklist...
Checklist file is complete.

Generating options file...
Options file is complete
*****
*   To complete your SMTP-MR configuration, carry out the steps
*   detailed in the setup checklist  MULTINET:SMTP-MR.CHECKLIST.
*****
$

```

## Configuring SMTP/MR Document Conversion

The DCF document conversion utility is used by SMTP/MR to convert WPS and DX message bodyparts to ASCII text. This utility is built from document conversion library routines supplied as part of the Message Router VMS MAIL Gateway (MRGATE) distribution kit. SMTP/MR can function without this utility, but cannot convert WPS and DX bodyparts to ASCII without it. Bodyparts in outbound SMTP mail that cannot be converted to ASCII are discarded.

The DCF utility is not supplied in executable form with SMTP/MR; it must be built after SMTP/MR is configured. The command procedure MULTINET:DCF\_BUILD.COM is provided to accomplish this. This procedure prompts for two items:

- The location of the save set from which to extract the necessary conversion libraries
- The name of a directory into which the libraries should temporarily be placed while the utility is being linked. Under MRGATE V3.1, the name of the save set containing the conversion libraries is MRGATE031.A.

You need not copy the saveset from the distribution media for DCF\_BUILD.COM to work. For example, if you want to access the libraries on a TK50 containing MRGATE while on a VAXstation 3100, you would specify the saveset name as MKA500:MRGATE031.A. The following example shows how to build the DCF utility from a saveset located in the SYS\$MANAGER directory:

```

$ @MULTINET:DCF_BUILD
Directory to put libraries in [SYS$SCRATCH]: RETURN
File specification of save set from which to extract libraries:

```

### **SYS\$MANAGER:MRGATE031.A**

```
Extracting libraries...
%BACKUP-S-CREATED, created SYS$SYSROOT:[SYSMGR]KOALA.OLB;1
%BACKUP-S-CREATED, created SYS$SYSROOT:[SYSMGR]DCF_BASE.OLB;1
%BACKUP-S-CREATED, created SYS$SYSROOT:[SYSMGR]DCF_TRANSLATE.OLB;1
%BACKUP-S-CREATED, created SYS$SYSROOT:[SYSMGR]DCF_MAIL_CONVERSIONS.OLB;1
%BACKUP-S-CREATED, created SYS$SYSROOT:[SYSMGR]DCF_DSAF.OLB;1
%BACKUP-S-CREATED, created SYS$SYSROOT:[SYSMGR]WPADOC.OLB;1
%BACKUP-S-CREATED, created SYS$SYSROOT:[SYSMGR]WPABASE.OLB;1
%BACKUP-S-CREATED, created SYS$SYSROOT:[SYSMGR]XPORT.OLB;1
Linking DCF utility...
Cleaning up...
Done
$
```

The DCF utility is never invoked interactively. It is always invoked automatically by the SMTP/MR gateway whenever it has mail containing WPS or DX bodyparts it needs to send via SMTP.

**Note:** You must use the A save set from MRGATE V3.1 or V3.2 to build DCF. Versions V3.3 and later do not contain the object files needed to link to DCF. If you upgrade to MRGATE V3.3, save your V3.1 or V3.2 distribution media.

## Completing SMTP/MR Configuration

In addition to the SMTP/MR configuration data files, the file `MULTINET:SMTP-MR.CHECKLIST` is created by the `MR_CONFIGURE.COM` command procedure. This file contains the steps needed to complete the SMTP/MR configuration, which include:

1. Adding the SMTP/MR gateway mailbox to your Message Router configuration. Run the `MRMAN` utility exactly as documented in the checklist file. The Message Router mailbox name and password must be exactly the same as you specified to `MR_CONFIGURE.COM`.
2. Building the WPS and DX document conversion utility. See the previous section for details on building this utility.
3. Configuring your Domain Name System (DNS) name server for SMTP/MR operation. You must add a Mail eXchanger (MX) record to your name server configuration for the following:
  - The domain name of the gateway itself (MR.EXAMPLE.COM in the example in the *Error! Reference source not found.* section)
  - The domain name used to direct mail to your ALL-IN-1 IOS users (A1.EXAMPLE.COM in the example in the *Configuring SMTP/MR* section)

- The domain name used to direct mail to your ALL-IN-1 MAIL users (AM.EXAMPLE.COM in the example in the *Configuring SMTP/MR* section).

If the host running SMTP/MR is named MIGUEL.EXAMPLE.COM, the DNS Resource Records (RRs) you would use in the DNS configuration file for the domain EXAMPLE.COM are:

```
MR.EXAMPLE.COM.  IN  MX  0  MIGUEL.EXAMPLE.COM.
A1.EXAMPLE.COM.  IN  MX  0  MIGUEL.EXAMPLE.COM.
AM.EXAMPLE.COM.  IN  MX  0  MIGUEL.EXAMPLE.COM.
```

For more detailed information on configuring a DNS name server, see the *Host Tables and DNS* chapter.

4. If you are not running a DNS name server locally, you must add additional host records to the MULTINET:HOSTS.LOCAL file for the host names of the gateway and your ALL-IN-1 users. Using the names from the above example, and assuming that the IP address for MIGUEL.EXAMPLE.COM is 128.0.0.1, you would add the following lines to MULTINET:HOSTS.LOCAL:

```
HOST : 128.0.0.1 : MR.EXAMPLE.COM : : : :
HOST : 128.0.0.1 : A1.EXAMPLE.COM : : : :
HOST : 128.0.0.1 : AM.EXAMPLE.COM : : : :
```

You should:

- Place these lines after the entry for MIGUEL.EXAMPLE.COM.
- Specify each name on a line by itself. Merging entries in the hosts file prevents the gateway from functioning properly.
- Recompile and re-install the host tables after adding the new entries.

For detailed information on adding entries to your host tables, see Chapter 10.

5. Submitting the command procedure MULTINET:MR\_TO\_MULTINET.COM to the appropriate batch queue on your system. This command procedure runs periodically to transfer mail from the SMTP/MR Message Router mailbox (normally SMTP) to the MultiNet mailer. Examine this command procedure before submitting it to ensure it runs in the batch queue and under the desired user name.

## Configuring the SMTP-DECnet Mail Gateway

MultiNet can be set up as a gateway to route mail between SMTP and DECnet-only hosts, with appropriate address translations to make the DECnet-style addresses easier for Internet hosts to interpret.

To do this, you set the `DECNET-DOMAIN` mail parameter and add an appropriate `MX` record to the Domain Name Service. The addresses of DECnet mail sent out via SMTP will be rewritten such that the DECnet node name(s) appear under the `DECNET-DOMAIN` name in the host-part of the address. The addresses of incoming SMTP mail for hosts under the `DECNET-DOMAIN` are automatically converted into DECnet addresses and delivered to the DECnet-only hosts.

## DECnet-to-SMTP Mail

In the DECnet-to-SMTP direction, a VMS MAIL user on a DECnet-only host sends SMTP mail by specifying an address of the form:

```
node::SMTP%"user@host"
```

`node` is the DECnet node name of the system running MultiNet.

MultiNet recognizes that the mail originated in DECnet and, if the `DECNET-DOMAIN` parameter is set, rewrites the originating address in the form

```
user@node.decnet-domain.
```

For example, `EXAMPLE.COM` has set up node `HQ` as a DECnet-SMTP gateway. A user named `JOHN` on DECnet-only node `WHARFIN` at `EXAMPLE.COM` addresses mail to the Info-MultiNet mailing list as: `HQ::SMTP%"Info-MultiNet@PROCESS.COM"`

`JOHN`'s DECnet return address, `WHARFIN::JOHN`, is rewritten by the gateway as:

```
JOHN@WHARFIN.DNET.EXAMPLE.COM
```

instead of:

```
"WHARFIN::JOHN"@HQ.EXAMPLE.COM
```

which some Internet mailers would have trouble parsing.

## SMTP-to-DECnet Mail

For the SMTP-DECnet gateway to work in the SMTP-to-DECnet direction, other hosts on your network must be told that mail for host names under the `DECNET-DOMAIN` should be sent to the gateway host. If you use Domain Name Service, the easiest way to do this is to set up a wildcard `MX` record for the `DECNET-DOMAIN`. In our example, the `MX` record looks like this:

```
*.DNET.EXAMPLE.COM. IN MX 0 HQ.EXAMPLE.COM.
```

This `MX` record causes other hosts on the Internet to send mail destined for any host under `DNET.EXAMPLE.COM` to node `HQ`. The gateway automatically recognizes the `DECNET-DOMAIN` in the host-name part of the address, rewrites the address to its DECnet form, and sends it through VMS MAIL.

If you do not use DNS, you must add a fully qualified host name for each DECnet node to your host tables. In our example, a return message to user `JOHN` on node `WHARFIN` would be addressed to:

JOHN@WHARFIN.DNET.EXAMPLE.COM.

When HQ receives that message, it translates the address to its DECnet form:

WHARFIN::JOHN

and sends the message to that address using VMS MAIL.



# 13. Printer Configuration

This chapter describes the printing services provided by MultiNet. MultiNet supports printing through the LPD, IPP and telnet (STREAM and NTY) protocols.

## Configuring the LPD/LPR Server

An LPD server allows other hosts and networks to access queues on the server system. The queues can be any valid OpenVMS queue, including LAT queues and terminal queues, and do not have to be MultiNet print client queues. A MultiNet system is disabled as an LPD server by default in the distribution kit.

To grant access to other hosts or networks, run the MultiNet Server Configuration Utility (SERVER-CONFIG) with the command:

```
$ MULTINET CONFIGURE /SERVERS
```

and enable the LPD server:

```
SERVER-CONFIG>ENABLE LPD
```

then select the LPD server:

```
SERVER-CONFIG>SELECT LPD
```

To add hosts, type the command:

```
SERVER-CONFIG>SET ACCEPT-HOSTS
```

You are first prompted to delete the hosts currently in the configuration, then prompted to add new hosts to the configuration by their IP addresses. To add access to all hosts on a particular network, issue the command:

```
SERVER-CONFIG>SET ACCEPT-NETS
```

You are prompted in a similar fashion for networks. For additional information on granting and denying access to MultiNet services, see Chapter 12.

Access control changes take effect immediately when you restart the MultiNet master server process with the RESTART command before exiting. To log errors to OPCOM, type this command before restarting the master server process:

```
$ REPLY/ENABLE=NET/TEMP
```

The following example shows how to grant LPD access to two specific hosts and to all hosts on a given network. (Note that HOSTS.EQUIV is not consulted when determining trusted hosts.)

```

$ MULTINET CONFIGURE /SERVERS
MultiNet Server Configuration Utility 5.6
[Reading in configuration from MULTINET:SERVICES.MASTER_SERVER]
SERVER-CONFIG>SHOW LPD /FULL
Service "LPD":
    TCP socket (AF_INET,SOCK_STREAM), Port 515
    Socket Options = SO_KEEPALIVE
    INIT() = TCP_Init
    LISTEN() = TCP_Listen
    CONNECTED() = TCP_Connected
    SERVICE() = Run_Program
    Program = "MULTINET:SERVER_LPD.EXE"
    Accept Hosts = IP*127.0.0.1
    Reject by default all other hosts and nets
    Reject Message = "Your host does not have line printer access"

SERVER-CONFIG>SELECT LPD
[The Selected SERVER entry is now LPD]
SERVER-CONFIG>SET ACCEPT-HOSTS
Delete address "IP*127.0.0.1" ? [NO]
You can now add new addresses for LPD.  An empty line terminates.
Add Address: 10.41.228.1
Add Address: 10.41.228.65
Add Address: RETURN
SERVER-CONFIG>SET ACCEPT-NETS
You can now add new addresses for LPD.  An empty line terminates.
Add Address: 10.16.72.0
Add Address: RETURN
SERVER-CONFIG>RESTART
Configuration modified, do you want to save it first ? [YES] RETURN
[Writing configuration to
MULTINET_COMMON_ROOT:[MULTINET]SERVICES.MASTER_SERVER]
%RUN-S-PROC_ID, identification of created process is 202002BD
SERVER-CONFIG>EXIT
[Configuration not modified, so no update needed]
$

```

## Setting a Default LPD User Name

A print request expects to be printed under a valid OpenVMS user name. If the print request originates from a remote system, the user name may not map to a valid OpenVMS user name. When defining a default LPD user name, any print request that does not map to a valid OpenVMS user name maps to the default LPD user name, and is accepted by the LPD server.

It is recommended that you define the default LPD user name even if the user names on the remote system match those on the OpenVMS system. Some printing systems use the name `root` or `lpd`

instead of the actual user name when dispatching a print job to a remote system; those names may not match any valid OpenVMS user name. To set a default LPD user name:

```
$ MULTINET CONFIGURE
MultiNet Network Configuration Utility 5.6
[Reading in MAXIMUM configuration from MULTINET:MULTINET.EXE]
[Reading in configuration from MULTINET:NETWORK_DEVICES.CONFIGURATION]
NET-CONFIG>SET LPD-DEFAULT-USERNAME PYEWACKET
NET-CONFIG>EXIT
[Writing configuration to MULTINET:NETWORK_DEVICES.CONFIGURATION]
[Writing Startup file MULTINET:START_MULTINET.COM]
[Changes take effect after the next VMS reboot]
```

To make the default user name take effect immediately without rebooting the system, define the following logical:

```
$ DEFINE/SYSTEM/EXECUTIVE MULTINET_LPD_DEFAULT_USERNAME "PYEWACKET"
```

**Note:** The LPD default user name must exist in the UAF file. It can be non-privileged and it can be "dis-used," but it must exist.

You can set up logical names to map remote LPD users to local users through a mechanism known as *proxy access*. Using logical names is useful when you want to receive LPD print jobs from a UNIX system on which the user names and UIDs on the client and server are completely uncoordinated.

Proxy mappings take precedence over the default mapping. To map a remote user's LPD requests for printing as if they were queued by a local user:

```
$ DEFINE/SYSTEM/EXECUTIVE MULTINET_LPD_PROXY 'user' 'local_user'
```

For example, to map remote user BROWN's LPD requests for printing as if they were queued by the local user JONES:

```
$ DEFINE/SYSTEM/EXECUTIVE MULTINET_LPD_PROXY BROWN JONES
```

## Changing the LPD Spool Directory

By default, LPD print jobs (and SMTP mail messages) on the OpenVMS system are stored in the directory `MULTINET_COMMON_ROOT:[MULTINET.SPOOL]`. You can change the directory with the `NET-CONFIG SET SPOOL-DIRECTORY` command:

```
$ MULTINET CONFIGURE
NET-CONFIG>SET SPOOL-DIRECTORY DISK$TEMP:[MULTINET]
```

You must redefine the logical that points to the spooling area unless you reboot the system after modifying the MultiNet configuration. Make sure the directory protections are set to `SYSTEM:RWED`, `OWNER:RWED`, `GROUP:RE`, and `WORLD:RE`.

```
$ DEFINE/SYSTEM/EXEC MULTINET_SPOOL_DISK$TEMP: [MULTINET]
```

## Cancelling LPD Print Jobs

MultiNet includes an implementation of the UNIX `lprm` utility which you can invoke with the `MULTINET LPRM` command. `LPRM` removes print jobs from remote UNIX (LPD) print queues and can be run by users.

Specify the jobs to be removed by job number. If you do not specify any jobs, the currently active job is deleted by default. The remote LPD queue associated with `SYS$PRINT` on the local system is used by default in the `MULTINET LPRM` command.

Use the `/QUEUE` qualifier to specify an alternate print queue, and the `/USER` qualifier to specify an alternate user name for the jobs to be deleted.

**Note:** You must have either `SYSPRV` or `OPER` privileges to specify a user name other than your own. To cancel all LPD print jobs, use the `/SUPERUSER` qualifier (which also requires `SYSPRV` privilege).

## Controlling Host Name Lookup

If the logical `MULTINET_LPD_DO_ASYNC_LOOKUP` is not defined to `1`, `True`, or `Yes`, then the LPD print symbiont will now resolve the remote host name with `getaddrinfo`, and is capable of resolving and connecting to an IPv6 address.

```
$ DEFINE/SYSTEM/EXEC MULTINET_LPD_DO_ASYNC_LOOKUP TRUE
```

# Configuring Printers on Remote Systems

Once you have set up the LPD server to accept incoming connections, you must set up the remote system for remote printing. When setting up print queues on other systems, specify the OpenVMS queue name as the remote printer name. (You can use the # character in remote printer names.) For example, an Apple LaserWriter is attached to the MultiNet host ABC.COM and is configured as the queue SYS\$PRINT. To configure a remote printer queue named "laser" on a UNIX system to direct output to SYS\$PRINT on EXAMPLE.COM, add the following entry to /etc/printcap:

```
laser|lw|LaserWriter II on host EXAMPLE.COM:\
      :rm=abc.com:rp=sys$print:sd=/var/spool/laser:lp=:
```

The following fields are required:

Name	Purpose
laser	Remote system queue name; in this example, laser
rm	Node name of the OpenVMS LPD server
rp	OpenVMS queue name on the OpenVMS LPD server
sd	Local UNIX spool directory
lp	Set to null to avoid LPD bugs

When a user on the UNIX system issues this command:

```
% lpr -Plaser foo
```

lpr searches /etc/printcap on the local system for an entry of "laser", and transfers the file "foo" to the LPD server on ABC.COM, which queues "foo" to SYS\$PRINT.

## Checking Remote Printer Queues

To display the contents of a remote queue that serves an OpenVMS print queue, use the command:

```
$ MULTINET SHOW /QUEUE=vms_queue_name
```

vms\_queue\_name is the name of the local OpenVMS queue.

The contents of the remote queue are accessed for display using the TCP LPD service.

**Note:** Queues configured with the STREAM protocol cannot be displayed with this command.

## LPD Jobs (Inbound)

MultiNet allows control of the job-queueing parameters used for incoming LPD print jobs using logical names. The formats for these logical names, which specify their scope, are:

*MULTINET\_LPD\_queue\_name\_typechar\_parametername*

Specifies a parameter value for a specific queue and file type.

*MULTINET\_LPD\_queue\_name\_\*\_parametername*

Specifies the default for a specific queue and all file types on that queue.

*MULTINET\_LPD\_\*\_typechar\_parametername*

Specifies the default for all queues with a specified file type.

*MULTINET\_LPD\_\*\_\*\_parametername*

Specifies a default for all queues and all file types.

Enter an asterisk in a logical name to match any *queue\_name* or *typechar*. An asterisk cannot be used for a *parametername*. The values in the logicals are:

- *parametername*, which is one of the following:

Value	Description
P1, P2, P3, P4, P5, P6, P7, P8	OpenVMS print job parameters [PRINT /PARAM= ( ) ]
BURST	/BURST qualifier (ALL, ONE, or NO)
FEED	/FEED qualifier (YES or NO)
FILETYPE	Type of spool file (FIXED512 or LFSTREAM)
FLAG	/FLAG qualifier (ALL, ONE, or NO)

FORM	Form name
PASSALL	/PASSALL qualifier (YES or NO)
SETUP	/SETUP qualifier, list of setup modules
TRAILER	/TRAILER qualifier (ALL, ONE, or NO)

- queuename, the name of a printer visible to LPD clients.
- typechar, a single letter specifying the file's data type; accepted values are c, d, f, g, l, n, p, r, t, and v. The f value is the lpr default if a user does not specify a type flag. r is required if a user specifies -f.

For example, if you want print jobs sent to your system via LPD and submitted to an OpenVMS print queue called PRINT01 to use the bottom input tray, you would issue this command:

```
$ DEFINE/EXEC/TABLE=MULTINET_PRINTER_TABLE -
_$ MULTINET LPD PRINT01 * P1 "INPUT TRAY=BOTTOM"
```

These logical names must all be defined in the MULTINET\_PRINTER\_TABLE logical name table. If you have not already run PRINTER-CONFIG, the MULTINET\_PRINTER\_TABLE logical name table may not exist. Verify the MULTINET\_PRINTER\_TABLE logical name table exists:

1. Start PRINTER-CONFIG with the following command:

```
$ MULTINET CONFIGURE /PRINTERS
```

2. Save the unmodified configuration:

```
PRINTER-CONFIG>SAVE
```

This command writes the file MULTINET:REMOTE-PRINTER-QUEUES.COM which contains the command to create the MULTINET\_PRINTER\_TABLE logical name table.

3. Exit PRINTER-CONFIG.

```
PRINTER-CONFIG>EXIT
```

4. Execute the MULTINET:REMOTE-PRINTER-QUEUES.COM command procedure.

**Note:** MULTINET:REMOTE-PRINTER-QUEUES.COM is executed automatically when MultiNet starts. If you define these logical names manually, they will be lost after rebooting. To preserve

your logical name definitions, add the appropriate `DEFINE` commands to your startup command procedure.

The following example shows how to configure the MultiNet LPD server to handle inbound jobs submitted with the `-v` option.

```
$ DEFINE/EXECUTIVE_MODE/TABLE=MULTINET_PRINTER_TABLE -  
_ $ MULTINET_LPD * V PASSALL "YES"
```

```
$ DEFINE/EXECUTIVE_MODE/TABLE=MULTINET_PRINTER_TABLE -  
_ $ MULTINET_LPD * V FILETYPE "FIXED512"
```

The following example specifies that all LPD jobs queued to `SYS$PRINT` be submitted with the form "WIDE":

```
$ DEFINE/EXECUTIVE_MODE/TABLE=MULTINET_PRINTER_TABLE -  
_ $ MULTINET_LPD SYS$PRINT * FORM WIDE
```

The `MULTINET_PRINTER_TABLE` logical name table and all logical names within the table must be defined in executive mode. To confirm this, enter the following command:

```
$ SHOW LOGICAL/FULL/TABLE=MULTINET_PRINTER_TABLE
```

`MULTINET_PRINTER_queue_name_PASSALL_FILTER` if a queue or a job is set to `/PASSALL`, this logical makes the default filter character "v" unless this logical exists and specifies a different value.

`MULTINET_PRINTER_queue_name_SUPPRESS_REMOTE_BANNER` if defined, does not include the "L" command in the control file to request a banner page.

`MULTINET_PRINTER_queue_name_NO_FFLF_DEFAULT` or

`MULTINET_PRINTER * NO_FFLF_DEFAULT` if the value is "Y", "T" or "1", then `NOFFLF` is enabled; otherwise it is disabled.

## Troubleshooting the LPD Server

The error message "record too large for user's buffer," indicates a file format problem. Define the following logicals, and specify that all print commands and any print qualifiers are treated as fixed 512-byte files.

```
$ DEFINE/EXECUTIVE_MODE/TABLE=MULTINET_PRINTER_TABLE -
```



```
_$ MULTINET LPD * * PASSALL TRUE
```

```
$ DEFINE/EXECUTIVE MODE/TABLE=MULTINET PRINTER TABLE -  
_$ MULTINET LPD * * FILETYPE FIXED512
```

For information about interpreting the asterisk (\*) in logical names, see the *LPD Jobs (Inbound)* section.

The file is removed from the queue on the client side, but the job never appears in the OpenVMS queue. Enable OPCOM with the command:

```
$ REPLY/ENABLE=NET/TEMP
```

and try the PRINT command again. The OPCOM error message, "your host does not have line printer access," indicates that the server has not been set up to allow incoming connections from the remote host. See the *Configuring the LPD/LPR Server* section.

If no OPCOM error messages display, use the following command to verify that the default LPD user name has been defined in EXECUTIVE mode:

```
$ SHOW LOGICAL/FULL MULTINET LPD DEFAULT USERNAME
```

If the logical does not exist, or is not defined in executive mode, refer to the *Setting a Default LPD User Name* section.

To generate a log file with LPD, do the following:

```
$ DEFINE/SYSTEM MULTINET LPD SYMBIONT DEBUG
```

A value is not required for the logical. If the logical exists, debug logging is ON; if the logical does not exist, debug logging is OFF. The equivalence string for the logical is not checked.

Restart the queue. If the OPCOM messages are enabled, a notification is printed telling you that debugging is enabled.

To turn off the SYMBIONT log file, deassign the logical and restart the SYMBIONT queue(s) that were (re)started since you defined the logical because the symbiont only looks for the logical when the queue starts up.

```
$ DEASSIGN/SYSTEM MULTINET LPD SYMBIONT DEBUG
```

The log files are created in MULTINET\_SPOOL:LPD\_DEBUG\_pid.LOG

pid is the hexadecimal process ID of the symbiont process in question. There may be as many as 16 queues sharing the same log file, since log files are per symbiont process, not per queue, and each process can support processing for up to 16 queues.

The error message Invalid data file size received, indicates a LPR client sent an invalid printing job. This could occur if a Windows 2000 LPR client is wrongly configured.

# Configuring Print Queues

This section describes how to configure a TCP/IP print queue using MultiNet. MultiNet supports two printer protocols: STREAM and LPD. Set up print queues using the interactive MultiNet `PRINTER-CONFIG` utility, which is used to configure OpenVMS TCP/IP printer queues.

`PRINTER-CONFIG` creates and maintains configuration information about MultiNet TCP/IP queues. By default, this utility stores configuration information in the file `MULTINET:REMOTE-PRINTER-QUEUES.COM`, which also contains the necessary DCL commands to configure the printer pseudo-devices and set up the OpenVMS print queues.

After using this utility to configure the queue, invoke the command procedure `MULTINET:REMOTE-PRINTER-QUEUES.COM` to create an OpenVMS queue on the system. You can invoke this command procedure at any time to initialize a new queue without affecting any queues already running. Use this queue exactly like any other OpenVMS queue; it responds to standard OpenVMS `PRINT` and `QUEUE` commands. To run the printer queue configuration manager, issue the command:

```
$ MULTINET CONFIGURE/PRINTERS
```

## Configuring an LPD Protocol Queue

The LPD protocol is a standard for sharing printers between hosts on an IP network. The protocol is defined by RFC-1179, and is integrated into the OpenVMS printing system with the `MULTINET_LPD_SYMBIONT` executable. LPD was originally intended for system-to-system print job transfers, but many printer Ethernet cards now support LPD protocol so that LPD can be used for printing directly to a printer. The following parameters are used with the `MULTINET_LPD_SYMBIONT`:

- `ADDRESS=host_addr` or `ADDRESS:host_addr`
- `CLASS=class_string` or `CLASS:class_string`
- `FILTER=filter_char` or `FILTER:filter_char`
- `NOFFLF=Y/T/1` or `NOFFLF:Y/T/1`
- `PRINTER=remote_queue_name` or `PRINTER:remote_queue_name`
- `RETAIN_CR=Y/T/1` or `RETAIN_CR:Y/T/1`

This protocol separates the print job into two parts:

- A `df` file (data file) that contains the actual data in the print file
- A `cf` file (control file) that contains commands for how the file should be printed

The df file is sent first and must be spooled on the server until the cf file arrives with instructions for how the server should print the file. This process causes more overhead when printing a file.

If you are using the LPD protocol to print directly to a printer, make sure the printer's network interface supports LPD. Consult the printer interface documentation to confirm this. To initially configure an LPD queue, use the `PRINTER-CONFIG ADD` command. The utility prompts you for the following information:

### Remote Host Name:

The IP address of the remote system or printer to which the print job will be sent. The address may be specified in standard "dot" notation (i.e. 123.456.789.012), or as a DNS name if there is an entry in the DNS server or host table of the sending system for the printer or terminal server in question.

### Protocol Type: [LPD]

Press **RETURN** to accept the default (LPD protocol).

### Remote Queue Name: [lp]

A remote queue name used by the LPD protocol. If this is a system-to-system print setup, the remote queue name must match the name of an existing queue defined on the remote system. If the print jobs will go directly to a printer, the remote queue name often designates the "spooling" area that exists on the printer's network interface card.

The remote printer queue name is not arbitrary; it is specific to the type of network interface installed in the printer. Consult the printer's network interface documentation for the correct remote queue name.

The following example shows the initial configuration of a queue called HP5 that points to an HP4si printer with an internal HP Jet Direct Card, using LPD protocol.

```
$ MULTINET CONFIGURE/PRINTER
MultiNet Remote Printer Configuration Utility 5.6
[Reading in configuration from MULTINET:REMOTE-PRINTER-QUEUES.COM]
PRINTER-CONFIG>ADD HP5
[Adding new configuration entry for queue "HP5"]
Remote Host Name: 10.49.2.3
Protocol Type: [LPD] Return
Remote Queue Name: [lp] TEXT
[HP5 => 10.49.2.3, TEXT]
PRINTER-CONFIG>SHOW HP5
Queue Name      IP Destination      Remote Queue Name
-----
HP5             10.49.2.3           TEXT
PRINTER-CONFIG>EXIT
[Writing configuration to MULTINET:REMOTE-PRINTER-QUEUES.COM]
```

Invoke the `MULTINET:REMOTE-PRINTER-QUEUES.COM` command procedure to initialize the printer queue.

## Input Record Modification

By default, the `MULTINET_LPD_SYMBIONT` does no input record modification. Records are read in, formatted and presented to the output code by the standard OpenVMS printer symbiont library routines. The LPD protocol is implemented in the output routine of the symbiont. There are times, however, when it may be desirable to handle record input differently. For instance, the standard routines treat records with leading form feed (FF) characters differently from other records. Leading and trailing carriage control is stripped from such records, and, if the record consisted only of a FF character, the leading carriage control bytes of the following record are suppressed. In most cases this yields the desired output on paper, but when it doesn't, it may be useful to be able to specify other behavior. That's what the LPD symbiont's `EMBED_CC` and `ADD_EOR` capabilities are intended to help with.

**Note:** Please see the *OpenVMS Utility Reference Manual* for more information about print symbionts, input filtering and the handling of various file formats.

The `EMBED_CC` and `ADD_EOR` capabilities are optional, and must be enabled through the use of a new logical name (`MULTINET_LPD_SYMBIONT_FILTER_ENABLED`) to be usable. Without this logical being present in the system logical name table at the time the symbiont process is started, neither of these capabilities will work. This logical causes the symbiont process to initialize differently than it does when the logical is not present. All other symbiont capabilities work normally regardless of the presence or absence of this logical.

The `EMBED_CC` capability allows the carriage control specified for the file to be embedded in the data records prior to their being passed to the main formatting routine of the OpenVMS print symbiont. The records will have no separate carriage control specified once this is done, and will be equivalent to printing an embedded carriage control file. The carriage control specified for each record can be the default provided by the input routines, or it can be specified on a system, queue, or job basis. If the carriage control is specified, this will override whatever implicit carriage control is present in the file being printed.

The `ADD_EOR` capability allows specification of a byte to be added to the end of each data record. This byte will follow any leading carriage control bytes, the data, and any trailing carriage control bytes if these are embedded, or precede them if they are not.

The `EMBED_CC` and `ADD_EOR` capabilities can be used individually or together, or not used at all, as needs dictate. Both require that the `MULTINET_LPD_SYMBIONT_FILTER_ENABLED` logical be defined when the symbiont process is first started.

Logical names are used to control these two capabilities on a system-wide or queue-specific basis. `PRINT` command parameters are used to control them on a per-job basis. See the appropriate sections below for more information on using these capabilities.

## Logicals used in controlling `EMBED_CC` and/or `ADD_EOR` operations:

### `MULTINET_LPD_SYMBIONT_FILTER_ENABLED`

Define as 1/T/Y to enable `EMBED_CC` and `ADD_EOR` capabilities.

Example:

```
$ DEFINE/SYSTEM/EXECUTIVE_MODE MULTINET LPD SYMBIONT FILTER ENABLED Y
```

### `MULTINET_LPD_EMBED_CC`

#### `MULTINET_LPD_queue_EMBED_CC`

Turns on carriage control (CC) embedding. Value is the specification for the CC longword if it's an 8-digit HEX number, beginning with 0x. If the value is 1/T/Y it enables embedding of the default CC as handed to the `Input_File` routine. If the value is 0/F/N, embedding is disabled.

The 8-digit HEX number used to specify an override CC value consists of four bytes, each of which specifies a different aspect of the CC. From lowest order byte (right most) to highest:

- Byte 1: Repeat count for Leading CC
- Byte 2: Leading CC value
- Byte 3: Repeat count for Trailing CC
- Byte 4: Trailing CC value

The CC values are the actual ASCII character code value, except that zero is not "NUL", but is used to specify the sequence "CRLF".

Example:

```
$ DEFINE/SYSTEM/EXECUTIVE_MODE MULTINET_LPD_EMBED_CC "0x0D010A01"
```

In the preceding example, the four bytes are shown from the highest order byte (fourth) to the lowest order byte (first), with “0D” representing the fourth byte, “01” representing the third byte, “0A” representing the second byte, and “01” representing the first byte. This value would specify a single leading Line Feed (LF) byte, and a single trailing Carriage Return (CR) byte.

```
$ DEFINE/SYSTEM/EXECUTIVE_MODE MULTINET_LPD_FOobar_EMBED_CC "Y"
```

says to embed the default CC for each record, as specified in the OpenVMS Utility Routines manual. For a Carriage-Return Carriage Control file this is a leading LF, trailing CR, with special handling around FF characters.

**Note:** MULTINET\_LPD\_SYMBIONT\_FILTER\_ENABLED must be defined when the symbiont process is started in order for this to be effective!

**MULTINET\_LPD\_ADD\_EOR**

**MULTINET\_LPD\_queue\_ADD\_EOR**

Specifies that an EOR marker byte is to be added to the end of each input record, and what the value of that marker is to be. The marker is specified as either a two-digit HEX number, prefixed with "0x", or the actual ASCII character to use.

Example:

```
$ DEFINE/SYSTEM/EXECUTIVE_MODE MULTINET_LPD_ADD_EOR "0x7C"
```

```
$ DEFINE/SYSTEM/EXECUTIVE_MODE MULTINET_LPD_ADD_EOR "|"
```

These two specifications are equivalent, and will result in a vertical bar, or “pipe”, character being added to the end of each record. If CC is being embedded, this byte will appear following the trailing CC byte(s). If CC is not being embedded, this byte will precede any trailing CC byte(s).

**Note:** MULTINET\_LPD\_SYMBIONT\_FILTER\_ENABLED must be defined when the symbiont process is started in order for this to be effective!

## Print parameters used in controlling EMBED\_CC and/or ADD\_EOR operations:

**/PARAMETER=(EMBED\_CC=*value*)**

The job parameter equivalent of MULTINET\_LPD\_EMBED\_CC. Value has the same range of options as for that logical.

**/PARAMETER=(ADD\_EOR=*value*)**

The job parameter equivalent of MULTINET\_LPD\_ADD\_EOR. Value has the same range of options as for that logical.

**Note:** MULTINET\_LPD\_SYMBIONT\_FILTER\_ENABLED must be defined when the symbiont process is started in order for either of these to be effective!

## Logical Names Provided for Controlling LPD Print Processing

Three logical names allow you to control LPD queue print job handling. By default, the LPD symbiont passes the letter "f" (for "formatted" file) as the print filter character the LPD server uses when you issue a standard PRINT request to an LPD queue. If you include the /PASSALL qualifier on your print request, the LPD symbiont uses the letter "v" (for "Versatec", that is, binary output) instead. In addition, the LPD symbiont normally strips CR (carriage return) characters from CRLF (carriage return, line feed) line-termination sequences, in keeping with the UNIX notion of LF (Line Feed) as the new-line character.

**MULTINET\_PRINTER\_queue\_name\_DEFAULT\_FILTER**

Allows you to specify the print filter character to use instead of "f" on the specified LPD queue. Declare the alternative, normal print filter character as follows:

```
$ DEFINE /TABLE=MULTINET_PRINTER_TABLE -  
_ $ MULTINET_PRINTER queuename DEFAULT_FILTER "character"
```

*queuename* is the name of the queue for which you are modifying the print filter character, and "*character*" is the character to be used.

You may override both the default character and the alternative character by including the /PARAMETER=(FILTER="*character*") qualifier on your print request (assuming the logical name MULTINET\_PRINTER\_*queuename*\_ALLOW\_USER\_SPEC is defined appropriately).

#### **MULTINET\_PRINTER\_*queuename*\_ALLOW\_USER\_SPEC**

If defined with an equivalence string of "Y", "T" or "I", the user can specify the ADDRESS and PRINTER values for the print job on the PRINT command, in the /PARAMETERS qualifier. In addition, specifying this logical will result in requeue retries, rather than the default timed retries when there is a problem with a connection. If this logical is not defined, or is defined with some other equivalence string, the ADDRESS and PRINTER parameters entered on the PRINT command by the user will be ignored, and timed retries will be performed.

#### **MULTINET\_PRINTER\_*queuename*\_PASSALL\_FILTER**

Allows you to specify the print filter character to use instead of "v" on the specified LPD queue. Declare the alternative passall print filter character as follows:

```
$ DEFINE /TABLE=MULTINET_PRINTER_TABLE -  
_ $ MULTINET_PRINTER queuename PASSALL_FILTER "character"
```

*queuename* is the name of the queue for which you are modifying the passall print filter character, and "*character*" is the character to be used.

You may override both the default character and the alternative passall character by including both the /PARAMETER=(FILTER="*character*") qualifier and the /PASSALL qualifier on your print request.

#### **MULTINET\_PRINTER\_*queuename*\_RETAIN\_CR\_DEFAULT**

Allows you to specify the disposition of CR characters in CRLF sequences on the specified LPD queue. Change CR processing on the queue as follows:

```
$ DEFINE /TABLE=MULTINET_PRINTER_TABLE -  
_ $ MULTINET_PRINTER queuename RETAIN_CR_DEFAULT "boolean"
```

*queuename* is the name of the queue for which you are modifying the CR disposition.



"*boolean*" indicates whether or not CR characters are to be retained in CRLF sequences included in text sent to the remote system for printing.

You may override both the default CR disposition and the alternative disposition by including the `/PARAMETER=(RETAIN_CR="boolean")` qualifier on your print request.

- The `MULTINET_LPD_SYMBIONT_DEBUG` logical name enables debug logging.
- The `MULTINET_PRINTER_queue_name_SUPPRESS_FF` logical name controls whether CRFF is added to jobs.
- The `MULTINET_PRINTER_queue_name_NO_TELNET` logical name controls Telnet IAC code expansion.

If you want an extra blank line on each page and, consequently, an extra blank page when the bottom margin has been reached, set the logical `MULTINET_NLPx_REMOTE_PRINTER` to include the configuration parameter `DOLFFF=Y`. Depending on your printer, it may be desirable to keep the behavior and not have the extra blank line and extra blank page.

## Using Retry Timers

When the symbiont cannot connect to the remote system, or the remote LPD server reports insufficient resources for printing a job, the symbiont requeues the job for a later attempt. Requeue attempts are reported directly to the user who submitted the print job only if the user specified `/NOTIFY` in the print command. The requeue time is controlled through logical names; you can control the length of time a job will wait before being attempted again after a connection failure by defining a logical name as follows:

```
$ DEFINE/SYSTEM MULTINET_LPD_SYMBIONT_RETRY_INTERVAL "delta-time"
```

The default value is "0 00:10:00.00", or ten minutes.

You can also control the maximum amount of time that should elapse before the symbiont gives up on a job with this command:

```
$ DEFINE/SYSTEM MULTINET_LPD_SYMBIONT_MAX_RETRY_INTERVAL "delta-time"
```

The default value is "0 02:00:00.00", or two hours.

You must specify the delta-time values in quotation marks, and with a space separating the number of days from the number of hours, so the symbiont can process them correctly.

LPD uses timer retries to queues that are not set up to allow user-specified printer destinations. For the other queues, jobs are placed in the queue to be tried again. The advantage to the timer retries is that

successive jobs are not sent to the printer until the symbiont can actually contact the printer. Use the following logical to control timer retry intervals:

```
$ DEFINE/SYSTEM/EXEC MULTINET_LPD_SYMBIONT_CONNECT_TIMERS "n n"
```

Its equivalence string should be two numbers, separated by one space, that specify a) the retry interval and b) the maximum retry time, in seconds. By default, the interval starts at 600 seconds (10 minutes); for each retry, that value is doubled until the maximum retry time of 7200 seconds (2 hours) is reached. The default would be represented by the following logical definition:

```
$ DEFINE/SYSTEM/EXEC MULTINET_LPD_SYMBIONT_CONNECT_TIMERS "600 7200"
```

- OPCOM messages from the LPD symbiont now include the queue name and entry number associated with the message.
- OPCOM messages from the LPD symbiont can be disabled by defining the following logical:

```
$ DEFINE/SYSTEM/EXEC MULTINET_PRINTER_NO_OPCOM true
```

It is recommended that the OPCOM messages not be disabled. They may block legitimate problem messages, in addition to informational messages about trying connections. This logical can be used for both the LPD and STREAM symbionts.

MULTINET\_LPD\_SYMBIONT\_LFTAIL and MULTINET\_LPD\_SYMBIONT\_\*\_LFTAIL allow reversion to legacy behavior of terminating jobs with an <LF> rather than <CR>. To enable this behavior, use one of these values: Y, T, or 1.

MULTINET\_LPD\_MAXSTREAMS specifies the maximum number of streams each symbiont process will handle.

MULTINET\_LPD\_KEEPAALIVE turns on keepalives when making socket connections. To enable this behavior, use one of these values: 1, y, Y, t, or T.

MULTINET\_LPD\_SYMBIONT\_RESOURCE\_TIMERS specifies the initial and maximum resource retry delay times.

## Adding Print Queue Parameters

You can define queue parameters on each MultiNet queue using the MULTINET CONFIGURE/PRINTER utility. Refer to the *MultiNet Administrator's Reference* for specific parameters. The following example shows how to add a device control library called HP3SI to the queue called HP5.

```
$ MULTINET CONFIGURE/PRINTER
```

```
MultiNet Remote Printer Configuration Utility 5.6
```

```
[Reading in configuration from MULTINET:REMOTE-PRINTER-QUEUES.COM]
```

```

PRINTER-CONFIG>SELECT HP5
[The Selected Printer is now HP5]
PRINTER-CONFIG>SET LIBRARY HP3SI
[Library HP3SI]
PRINTER-CONFIG>SHOW HP5
Queue Name          IP Destination          Remote Queue Name
-----
HP5                  10.49.2.3                TCP port 9100
Device Control Library = HP3SI
PRINTER-CONFIG>EXIT
[Writing configuration to MULTINET:REMOTE-PRINTER-QUEUES.COM]

```

Remember to invoke the `REMOTE-PRINTER-QUEUES.COM` procedure after exiting the configuration utility to add the parameters to the queue. Some parameters are only valid with the `INITIALIZE /QUEUE` command, so you may need to first stop and delete the queue, then invoke the `REMOTE-PRINTER-QUEUE` command procedure before the new parameters take effect.

## Starting Multiple Print Queues

You can start either a single print queue or a list of queues, in addition to starting all print queues using the command procedure `MULTINET:REMOTE-PRINTER-QUEUES.COM`. The syntax for the command procedure is:

```
$ @MULTINET:REMOTE-PRINTER-QUEUES [queue1, [queue2, queue3, . . .]]
```

If you do not provide any queue names on the command line, the procedure attempts to start all defined queues. For example, to start `SYS$PRINT`:

```
$ @MULTINET:REMOTE-PRINTER-QUEUES SYS$PRINT
```

Or, to start `SYS$PRINT` and `SYS$LASER`:

```
$ @MULTINET:REMOTE-PRINTER-QUEUES SYS$PRINT, SYS$LASER
```

## Using User-Specified Print Destinations

MultiNet allows users to override some of the parameters that you specify when you configure a print queue. You can enable user-specified print destinations, using the `ADDRESS` and `PRINTER` parameters, on a per-queue basis:

```
$ DEFINE/TABLE=MULTINET PRINTER TABLE -
_ $ MULTINET PRINTER queueName ALLOW USER SPEC TRUE
```

If you want to allow users to specify their own print destinations for LPD printing, define this logical name during your system startup sequence. To define the logical using `MULTINET CONFIG`:

```
$ MULTINET CONFIGURE /PRINTER
PRINTER-CONFIG>SELECT queuename
PRINTER-CONFIG>SET ALLOW-USER-SPECIFIC ENABLE
PRINTER-CONFIG>EXIT
```

To override the parameters associated with an LPD queue, use the /PARAMETERS qualifier with the PRINT command. Specify an alternative destination, a print filter, or how carriage return characters are to be treated with these parameters:

**ADDRESS=*n.n.n.n***

Specifies the address of the destination host. This value must be a numeric IP address, not a host name. If not specified, the address configured for the queue in the MultiNet printer configuration utility (PRINTER-CONFIG) is used.

**CLASS=*class\_string***

Specifies the string to put on the "class" line in the control file. This gets used in various ways, but mostly appears on the banner page if one is printed.

**"PRINTER=*name*"**

Specifies the name of the printer on the destination host. Use quotation marks around the entire parameter specification, since many systems are case-sensitive regarding printer names. If you specify ADDRESS and omit this parameter, the printer name defaults to "lp." If you omit both ADDRESS and PRINTER, the printer name configured for the queue in the MultiNet printer configuration utility is used.

**RETAIN\_CR={Y | N}**

Specifies whether the symbiont should not convert CR/LF sequences into bare LFs when sending a text file to the remote system. The default is N; CR/LF sequences are converted to bare LFs. A setting of Y means the sequences are not converted. This parameter is ignored if you submit the job with the /PASSALL qualifier.

**"FILTER=*x*"**

Specifies the "print filter" character to be sent with the print job to the remote system. Use quotation marks around the entire parameter, since filter specifications are generally case-sensitive. By default, the symbiont uses "f" for text files and "v" for files submitted with the /PASSALL qualifier.

## RESTART

Restarts the queues with the modified configuration information.

## NOFFLF=Y/T/1

Specifies whether the symbiont does not add a Line Feed after a Form Feed when sending a text file to the remote system.

For example, to print to an alternate destination:

```
$ PRINT/PARAM=(ADDRESS=192.168.34.22,"PRINTER=HP5",RETAIN=Y,FILTER="1") -
_$ LOGIN.COM
```

If the case of the printer name or filter character must be preserved, these parameter values must be enclosed in quotes. In the preceding example, the filter character has been specified as a lowercase "l".

In the example, the LPD client symbiont is instructed to send the file to the LPD server on host 192.168.34.22 for queue HP5, retaining carriage return characters, and using a filter of "l". The queue entry in the local OpenVMS LPD client queue looks like:

```
$ SHOW QUEUE/FULL SYS$PRINT
Printer queue SYS$PRINT, stopped, on NODE:NLP8, mounted form DEFAULT
 /BASE_PRIORITY=4, /DEFAULT=(FORM=DEFAULT) /OWNER=[SYSTEM]
 /PROCESSOR=MULTINET_LPD_SYMBIONT /PROTECTION=(S:M,O:D,G:R,W:S)
Entry  Jobname      Username      Blocks      Status
-----
795    LOGIN         USER          9           Printing
Submitted 23-MAY-2003 09:40 /FORM=DEFAULT
 /PARAM=("ADDRESS=192.168.34.22", "PRINTER=HP5", "RETAIN=Y", "FILTER=l")
 /PRIORITY=100
File: _DKA100:[USER]LOGIN.COM;1
```

## Customizing Printer Queues

You can add special characteristics to queues by using customized command procedures. You can do this globally to all STREAM or LPD queues, or limit them to each individual queue. These command procedures are automatically invoked every time REMOTE-PRINTER-QUEUES.COM is invoked.

A customized command procedure must contain all commands to spool the NLP device to the queue and to initialize the queue. Review the command procedure MULTINET:REMOTE-PRINTER-QUEUES.COM to see how and when these customized command procedures are called.

- To add a special queue characteristic or qualifier to all STREAM queues, create a command procedure called INITIALIZE\_STREAM\_QUEUE.COM in the MULTINET directory.

- To customize all LPD queues, create a file called `INITIALIZE_LPD_QUEUE.COM` in the `MULTINET` directory.
- To customize individual queues, create a file called `INITIALIZE_queueName.COM` in the `MULTINET` directory.

The following example shows a customized command procedure to add the `/SEPARATE` qualifier to a queue called `HP5`. The file is located in the `MULTINET` directory and is called `INITIALIZE_HP5.COM`.

```

$! Custom initialization procedure for stream queue HP5
$!
$ NLP_Device = P1
$ Remote_Address = P2
$ Remote_port = P3
$ Default_Form = P4
$!
$ If F$GetDVI(NLP_Device,"SPL") Then Set Device/NoSpool 'NLP_Device'
$ Set Device/Spool=HP5 'NLP_Device':
$ Initialize/Queue/Processor=MultiNet_Stream_Symbiont HP5 -
/On='NLP_Device'/Start/library=hp3sidevctl/separate=(reset=(reset))
$ Exit

```

**Note:** The customized command procedure must contain all the commands to spool the NLP.

The NLP device must be unique for each queue. The `MULTINET:REMOTE_PRINTER_QUEUES.COM` command procedure chooses the next available NLP device when initializing the queue. To determine the correct NLP device, use `PRINTER-CONFIG` to add a new queue. After exiting the utility, examine the `MULTINET:REMOTE_PRINTER_QUEUES.COM` command procedure to determine which NLP device was assigned to the new queue. Use the commands in that command procedure as a template for creating the customized command procedure.

**Note:** `REMOTE_PRINTER_QUEUES.COM` passes the NLP device name as one of several parameters to the customized command procedure.

# Configuring a STREAM Protocol Queue

The print client STREAM protocol is a TELNET-based protocol that is not defined by an RFC. This is the recommended protocol for printing from VAX systems and is integrated into the OpenVMS printing system with the `MULTINET_STREAM_SYMBIONT` executable.

To initially configure a STREAM queue, use the `PRINTER-CONFIG ADD` command. The `ADD` command has a maximum limit of 5000 symbionts per system. The utility prompts you for the following information:

**Remote Host Name:**

The IP address of the printer or of the terminal server to which the printer is attached. The address may be specified in standard "dot" notation (i.e., 123.456.789.012), or as a DNS name if there is an entry in the DNS server or host table of the sending system for the printer or terminal server in question.

**Protocol Type: [LPD]**

Type the word `STREAM` to indicate STREAM protocol.

**TCP Port Number: [23]**

The port number to which the TCP/IP connection will attach. If the printer has an Ethernet card installed in it, the port number is listed in the Ethernet card documentation; check there first, or call the card manufacturer to confirm the port number. For example, HPLJ Jet Direct cards use port 9100 for the TCP port number.

If the printer is attached to a terminal server, the port number refers to the physical port on the terminal server to which the printer is attached. For example, the HP 90TL terminal server adds 2000 to the actual physical port number to address the correct port. If the printer is physically attached to port 3, add 2000 to 3, making the port number 2003. Direct any questions on addressing the correct port number to the appropriate terminal server vendor.

The following example shows the initial configuration of a queue called `HP5` that points to an HP4si printer with an internal HP Jet Direct Card, using STREAM protocol. (For terminal servers, such as the HP 90TL, make sure the logical name `MULTINET_PRINTER_printrname_NO_TELNET` is enabled. where `printrname` is the name that was given to the printer in `MU CONFIGURE /PRINTERS.`)

```
$ MULTINET CONFIGURE/PRINTER
```

```
MultiNet Remote Printer Configuration Utility 5.6
```

```
[Reading in configuration from MULTINET:REMOTE-PRINTER-QUEUES.COM]
```

```
PRINTER-CONFIG>ADD HP5
```

```
[Adding new configuration entry for queue "HP5"]
Remote Host Name: 192.168.2.3 or printer.example.com
Protocol Type: [LPD] STREAM
TCP Port Number: [23] 9100
[HP5 => 192.168.2.3, TCP port 9100 (no telnet option negotiation)]
PRINTER-CONFIG>SHOW HP5
Queue Name IP Destination Remote Queue Name
-----
HP5 192.168.2.3 TCP port 9100
    Telnet Options Processing will be suppressed
PRINTER-CONFIG>EXIT
[Writing configuration to MULTINET:REMOTE-PRINTER-QUEUES.COM]
Invoke the MULTINET:REMOTE-PRINTER-QUEUES.COM command procedure to initialize the
printer queue.
```

## Troubleshooting a STREAM Protocol Queue

To generate a log file with STREAM, do the following:

```
$ DEFINE/SYSTEM MULTINET_STREAM_SYMBIONT_DEBUG
```

A value is not required here as with NTYSMB. If the logical exists, debug logging is ON; if the logical does not exist, debug logging is OFF. The equivalence string for the logical is not checked.

Restart the queue. If the OPCOM messages are enabled, a notification is printed telling you that debugging is enabled.

To turn off the STREAM log file, deassign the logical and restart the STREAM queue(s) that were (re)started since you defined the logical because the symbiont only looks for the logical when the queue starts up.

```
$ DEASSIGN/SYSTEM MULTINET_STREAM_SYMBIONT_DEBUG
```

The log files are created in `MULTINET_SPOOL:STREAM_DEBUG_pid.LOG`

`pid` is the hexadecimal process ID of the symbiont process in question. There may be as many as 16 queues sharing the same log file, since log files are per symbiont process, not per queue, and each process can support processing for up to 16 queues.

## Logical Names Provided for Controlling STREAM Processing

- The `MULTINET_STREAM_SYMBIONT_DEBUG` logical name enables debug logging.



- The `MULTINET_PRINTER_queue_name_SUPPRESS_FF` logical name controls whether CRFF is added to jobs.
- The `MULTINET_PRINTER_queue_name_NO_TELNET` logical name controls Telnet IAC code expansion.
- If `MULTINET_STREAM_DO_ASYNC_LOOKUP` is not defined to 1, True, or Yes, then the stream print symbiont will now resolve the remote host name with `getaddrinfo`, and is capable of resolving and connecting to an IPv6 address.
- The `MULTINET_STREAM_DEAD_LINK_TIMEOUT` and `MULTINET_STREAM_queue_name_DEAD_LINK_TIMEOUT` logical name control dead link detection and handling.

The `MULTINET_STREAM_SYMBIONT` can be configured for detection and handling of "dead links", that is, a TCP/IP link that stops responding, but was not closed properly by the remote end.

To enable dead link detection, define a system logical:

```
$ DEFINE/SYSTEM MULTINET STREAM DEAD LINK TIMEOUT "timeout_secs"
[[requeue_secs] option]"
```

or:

```
$ DEFINE/SYSTEM MULTINET STREAM queue_name DEAD LINK TIMEOUT "timeout_secs"
[[requeue_secs] option]"
```

Both of the `_secs` values are integers specifying seconds. `Timeout_secs` is the number of seconds a write has to take before it is considered a dead link. `Requeue_secs` specifies how many seconds to hold a job if it gets requeued due to a dead link. `Option` is one of the following:

- "REQUEUE" - requeue the job with a hold for `requeue_secs`.
- "CONTINUE" - continue with job...just open a new link.
- "STOP" - stop the queue.

The default behavior is "REQUEUE" if none is specified explicitly, and the default `requeue_secs` is zero (that is, no delay) if no time is specified.

**Note:** None of these guarantee that a print job will not be affected adversely by a lost link, especially when it is due to the printer interface being powered off suddenly.

A problem can occur when the symbiont tries to open a connection and the remote host refuses the connection. Possible reasons for this refusal could be that there is another connection already open on

that port by another system or the connection from the previous job the symbiont sent has not finished closing. You can control how long the symbiont waits to retry a connection after it is refused with the `MULTINET_STREAM_SYMBIONT_TIMERS` logical. The logical sets the initial and the maximum time to wait before retrying the connection. For example, if you defined the logical as follows it would retry the connection after 1 second and double the time between subsequent retries until it reached the maximum of 10 seconds.

```
$ DEFINE/SYSTEM/EXEC MULTINET_STREAM_SYMBIONT_TIMERS "1 10"
```

## LPD and Stream Symbiont User Exit Support

A user exit is a mechanism for customizing the way the print symbiont sends jobs to the printer, beyond the methods provided by MultiNet. A user exit is a shareable image that contains symbols the symbiont reads at runtime for special processing instructions.

The file `MULTINET_ROOT:[MULTINET.EXAMPLES]USER_LPD_CLIENT.C` (provided in the distribution kit) contains the template for an LPD symbiont customization image. Note that this interface is subject to change without notice. Process Software does not support customer modifications to these interfaces.

The file shows how to modify the way the LPD protocol symbiont generates the control file it sends over the network, and the name and Internet address of the server to which it connects.

**Note:** If you use user exits, you must update your source code, recompile, and relink your routines.

See the comments in the beginning of the file for information on compiling and linking the code properly. Install the routine by placing `USER_LPD_CLIENT.EXE` in the `MULTINET:` directory. The customized image will be used the next time the symbiont starts.

The Stream symbiont supports user exits provided in the `MULTINET_ROOT:[MULTINET.EXAMPLES]` directory (available only if the library and include files are installed during the MultiNet installation procedure). The Stream user exit is named `USER_STREAM_CLIENT.C`.

The LPD user exit takes the first parameter specified in the OpenVMS `PRINT /PARAMETER=qualifier` and maps it to the job classification (emulating the UNIX `lpr -C` option). After adding changes to an exit file, compile the file, link it with the `/SHARE` qualifier, and copy the resulting executable into the common directory `MULTINET_COMMON_ROOT:[MULTINET]`.

`USER_LPD_CLIENT.C` supports the addition of a linefeed after a formfeed. You use logical names and print parameters to control the behavior. Use a logical name of the form `MULTINET_PRINTER_queue_name_NO_FFLF_DEFAULT:`

*queue\_name* is the name of the queue you are using or is the `*` character to designate all LPD queues defined in the logical name table `MULTINET_PRINTER_TABLE` to execute each time MultiNet starts:

```
$ DEFINE/EXECUTIVE/TABLE=MULTINET_PRINTER_TABLE ...
```

or use the `NOFFLF` parameter in a print command to control the behavior for a particular queue:

```
$ PRINT/PARAMETERS=(NOFFLF=TRUE) filespec
```

The LPD and Stream symbionts have been enhanced in the following ways:

- The first 16 bytes of the work area (the per-stream data area) in the symbionts are reserved for customer use.
- The customizable `psm_max_streams_v33()` routine is called by the symbionts to determine the number of streams that a symbiont should initialize.
- International character set translations are supported with the shareable user exit image `MULTINET:USER_TRANSLATE.EXE`. After adding changes to a customizable file, compile the file and link it with the `/SHARE` qualifier. (You can request a sample `USER_TRANSLATE.C` source module from Process Software Technical Support.)
- All MultiNet LPD and stream symbionts can be configured with a remote printer or host's domain name in addition to its IP address.

## Using the NTYSMB Symbiont for Remote, TCP-Connected Printers

MultiNet provides a print symbiont for sending print jobs to remote TCP-connected printers. This symbiont – `NTYSMB` - which works in conjunction with network terminal port (NTY) devices, can be used instead of MultiNet `STREAM` print queues. The `NTYSMB` symbiont:

- Allows for user exits. See file `USER_NTYSMB_CLIENT.C` in the `MULTINET_ROOT:[MULTINET.EXAMPLES]` directory.

- Sends a form feed at the end of a job.
- Corrects timer handling in the case where the maximum timeout is reached. The timers are controlled by two values taken from the equivalence string for the MULTINET\_NTYSMB\_TIMERS logical name, *initial* and *ceiling*. The values for *initial* and *ceiling* are given in seconds. The *initial* value is how soon, after the first connection attempt fails, the symbiont is to retry the connection. On subsequent connection failures, the symbiont backs off its retries exponentially, until it's only retrying every *ceiling* seconds. By default, *initial* is 10 seconds and *ceiling* is 7200 seconds (2 hours). The NTYSMB symbiont never gives up on a job. Define the logical name as:

```
$ DEFINE/SYSTEM/EXEC MULTINET_NTYSMB_TIMERS "initial ceiling"
```

- Zeros out channel information in case a write request is received when a shutdown or close is in progress.
- Fixes queue shutdown when a timed retry is outstanding and a STOP/REQUEST is issued against the queue.
- Corrects I/O synchronization problems where data could be sent to the printer out of order.

MULTINET\_NTYSMB\_DEBUG causes various debug options to be enabled.

MULTINET\_NTYSMB\_\*\_MAXTIMERMSG and MULTINET\_NTYSMB\_queue\_name\_MAXTIMERMSG specifies the message to be issued when the connection timer hits the maximum value. One "%s" argument will be supplied in the form of the queue name.

## NTYSMB Advantages Over STREAM Queues

NTY queues have the following advantages over STREAM queues:

Improved print formatting	The standard OpenVMS print symbiont normally takes advantage of formatting capabilities in the OpenVMS terminal driver. These capabilities were only partially emulated by the STREAM symbiont. Because MULTINET_NTYSMB uses network terminal port devices, full print symbiont formatting is now supported.
More control over queues' devices	The standard SET TERMINAL and queue operation commands are used to set up the NTY terminal device and MULTINET_NTYSMB print queues. System managers wanting to take advantage of the OpenVMS queue manager's Autostart capability may now do so with this new print queue support.

Familiar management interface	The NTY Control Program (NTYCP) and <code>MULTINET_NTYSMB</code> print symbiont are patterned after Hewlett-Packard's LAT Control Program (LATCP) and LAT print symbiont, providing a management interface that should be familiar to many OpenVMS system managers.
-------------------------------	---

## Setting Up a Print Queue with `MULTINET_NTYSMB`

MultiNet must be started before NTY devices can be created or `MULTINET_NTYSMB` print queues can be initialized or started.

System managers using the OpenVMS queue manager's Autostart capability *must* leave Autostart *disabled* until after MultiNet is started and NTY devices have been set up.

To set up a print queue with `MULTINET_NTYSMB`:

1. Create the NTY device.

Use the NTY Control Program (NTYCP) to create the terminal device:

```
$ NTYCP := $MULTINET:NTYCP
$ NTYCP CREATE PORT NTYnnnn /NODE=node/port
```

You can invoke the NTYCP program as an OpenVMS "foreign" command, as shown above, or run in interactive mode:

```
$ RUN MULTINET:NTYCP
NTYCP>CREATE PORT NTYnnnn /NODE=node/port
NTYCP>EXIT
```

NTYCP uses standard DCL-style command parsing. The "?" help feature available in other MultiNet utilities is not available in NTYCP.

2. Set up the terminal characteristics:

```
$ SET TERMINAL NTYnnnn: /PERMANENT/NOBROADCAST/NOTYPEAHEAD/NOWRAP/FORM
```

3. Set up spooling, if desired:

```
$ SET DEVICE/SPOOLED=(queue-name, SYS$SYSDEVICE: ) NTYnnnn
```

4. Initialize and start the queue:

```
$ INITIALIZE/QUEUE/ON=NTYnnnn: queue/PROCESSOR=MULTINET_NTYSMB/START
```

The following is an example of a print queue set up to an HP LaserJet printer with a JetDirect card.

```
$ NTYCP := $MULTINET:NTYCP
$ NTYCP CREATE PORT NTY1001/NODE=hp-laserjet/PORT=9100
%NTYCP-S-CREPORT, device NTY1001: created to host 192.1.1.5, port 9100
$ SET TERMINAL/PERMANENT NTY1001:/NOBROADCAST/NOTYPEAHEAD/NOWRAP/FORM
$ INITIALIZE/QUEUE/ON=NTY1001: HP LASERJET/PROCESSOR=MULTINET NTYSMB/START
```

## Troubleshooting the MULTINET\_NTYSMB

To generate a log file with NTYSMB, do the following:

```
$ DEFINE/SYSTEM MULTINET_NTYSMB_DEBUG 7
```

The equivalence string for this logical is an integer. Various bits in this value control different aspects of debug logging. A value of 7 enables full debug logging, and is the recommended setting.

Restart the queue. If the OPCOM messages are enabled, a notification is printed telling you that debugging is enabled.

To turn off the NTYSMB log file, deassign the logical and restart the NTYSMB queue(s) that were (re)started since you defined the logical because the symbiont only looks for the logical when the queue starts up.

```
$ DEASSIGN/SYSTEM MULTINET_NTYSMB_DEBUG
```

The log files are created in MULTINET\_SPOOL:NTYSMB\_DEBUG\_*pid*.LOG

*pid* is the hexadecimal process ID of the symbiont process in question. There may be as many as 16 queues sharing the same log file, since log files are per symbiont process, not per queue, and each process can support processing for up to 16 queues.

Since these log files are intended to assist Process Software in deciphering problems, their output might not make sense to you.

## Troubleshooting the Print Queue

This section describes potential print queue problems and recommended solutions.

- When executing REMOTE-PRINTER-QUEUES.COM, the following messages may appear:

```
%SET-E-NOTSET, error modifying NLPx:
-SYSTEM-W-DEVASSIGN, device has channels assigned
%SET-E-NOTSET, error modifying NLPx
```

```
-CLI-E-DEVALSPL, device already spooled
%JBC-I-QUENOTMOD, modifications not made to running queue4
```

These messages occur when the command file tries to modify an existing queue. The messages are only warnings and do not affect the queues in any way.

- Another situation may occur where the print job is "stair-stepping" down the page. It appears as if the job contains carriage line feeds, but no carriage returns.
  - If the queue is set up as an LPD queue pointing to a printer, the wrong remote queue name has been specified in the printer configuration. Check the documentation for the printer's Ethernet card for the correct remote queue name. (See the *Configuring an LPD Print Queue* section.)
  - If the queue is set up as a STREAM queue pointing to a Hewlett-Packard LaserJet, the problem is the line termination setting on the printer. Most HPLJ printers are initially configured with line termination set to <LF>=<LF>. For printing from VMS systems, the line termination should be set to <LF>=<LF>+<CR>. Consult the printer documentation about changing this parameter.
- The print job prints all on one line.

If this is a STREAM queue pointing to a printer connected to a terminal server, the terminal server characteristics must be modified. Check the terminal server documentation for the proper commands and instructions for logging onto the terminal server. The following example shows the commands for a Hewlett-Packard 90TL terminal server.

```
CHANGE PORT port# TELNET SERVER NEWLINE TO HOST <CRLF>
CHANGE PORT port# TELNET SERVER NEWLINE FROM TERMINAL <CRLF>
```

- The queue prints text properly, but graphics do not print correctly.

If the TELNET negotiation option is ON, you may need to disable it with the SET SUPPRESS-TELNET parameter described in the *MultiNet Administrator's Reference*.

The default is to set TELNET negotiation OFF by default, unless it is using the default port 23. The following example shows how to disable TELNET negotiation.

```
$ MULTINET CONFIGURE/PRINTER
PRINTER-CONFIG>SEL queue-name
PRINTER-CONFIG>SET SUPPRESS-TELNET ENABLE
PRINTER-CONFIG>EXIT
```

Be sure all privileges are enabled; then invoke MULTINET:REMOTE-PRINTER-QUEUES.COM.

- PostScript files do not print.

When printing PostScript files, set the queue to NOFEED. Remove any FLAG pages from the queue. See the *MultiNet Administrator's Reference* for information about the SET NOFEED and

SET FLAG commands. You should also specify that the OpenVMS form on which the job is printed be set to the maximum width, and defined with /NOWRAP and /NOTRUNC. Use the following commands to determine the characteristics of the form being used on the queue:

```
$ SHOW QUEUE/FORM/FULL form-name
```

An example of the characteristics on a form called POSTSCRIPT is shown in the following example:

Form name	Number	Description
-----	-----	-----
POSTSCRIPT (stock=DEFAULT) /LENGTH=66 /MARGIN=(BOTTOM=6) /STOCK=DEFAULT /WIDTH=65535	3	Postscript input; white paper

To define the form with the proper characteristics, use the DEFINE/FORM command.

## Internet Printing Protocol (IPP)

The IPP print symbiont is an OpenVMS print symbiont working with the OpenVMS printing subsystem to implement an IPP Client. It allows printing over a network to printers and servers that support the IPP v1.0 network printing protocol. The user interface is similar to other print symbionts in that it uses PRINT commands or system library calls to submit jobs to print queues. The IPP protocol has specific qualifier values and queue settings that must be present to allow the symbiont to function. This section describes both the configuration of IPP print queues and the use of the PRINT command. For information on submitting jobs to print queues using system library calls, see the appropriate OpenVMS documentation.

## IPP Protocol Background

The Internet Printing Protocol solves a number of problems in existing network printing protocols; the most common is the LPD protocol, originally implemented on UNIX operating systems.

From RFC 2568:

"The Internet Printing Protocol (IPP) is an application level protocol that can be used for distributed printing on the Internet. This protocol defines interactions between a client and a server. The protocol allows a client to inquire about capabilities of a printer, to submit print jobs and to inquire about and cancel print jobs. The server for these requests is the Printer; the Printer is an abstraction of a generic document output device and/or a print service provider. Thus, the



Printer could be a real printing device, such as a computer printer or fax output device, or it could be a service that interfaced with output devices."

IPP has a better error reporting capability than LPD or TELNET. It supports multi-sided printing, landscape/portrait layouts, and multiple pages per physical sheet ("number-up") printing. Because not all printer models that support IPP will support all capabilities, the IPP protocol includes a way for the symbiont to query the printer as to its capabilities before a job is sent. If the printer cannot handle a given request, the job is aborted with an error status. The error status appears in the system accounting log.

IPP uses the HTTP 1.1 protocol as its transport layer; however, it has its own reserved port number, port 631. You can use the IPP Symbiont to print to other port numbers, including the standard HTTP port (80), but you need to specify the port number as part of the printer URL if the port number is not the default IPP port number. If you are printing through a firewall this could be a factor to consider. For a full description of the IPP protocol, see the relevant RFCs listed below.

The IPP Printer Working Group has a web page with at least a partial list of printers that claim to support IPP. It is at:

<http://www.pwg.org/ipp/index.html>

## Relevant RFCs

The RFCs related to IPP v1.0 are:

Design Goals for an Internet Printing Protocol	RFC 2567
Rationale for the Structure and Model and Protocol for the Internet Printing Protocol	RFC 2568
Internet Printing Protocol/1.0: Model and Semantics	RFC 2566
Internet Printing Protocol/1.0: Encoding and Transport	RFC 2565
Internet Printing Protocol/1.0: Implementer's Guide	RFC 2639
Mapping between LPD and IPP Protocols	RFC 2569

Additional RFCs are referenced by these, such as the ones describing HTTP v1.1, MIME Media Types, etc. The specific RFCs are called out in the above documents.

## Limitations of this Implementation

The IPP symbiont implements a subset of the IPP v1.0 protocol consisting of all required portions and several selected optional features. Note that not all features are available on all printers; most printers implement a subset of the available protocol capabilities.

Not all printers claiming to support IPP implement IPP correctly. Some use supersets of HTTP 1.0, rather than the required HTTP 1.1. Some do unusual things with TCP/IP connections, such as having extremely short timeouts. The symbiont has been adapted to support as many of these inconsistencies as possible (see the `EXPECT_LINK_CLOSURES` option for an example). The symbiont may or may not behave as expected with such printers depending on your particular network characteristics and exactly what the printer manufacturer has done differently from what is specified in the RFCs. The symbiont should work with any fully compliant IPP v1.0 printer or server.

## Configuration

The IPP symbiont has a flexible configuration. You can supply the information in the queue setup itself, as the `/DESCRIPTION=` string which is supported by OpenVMS as part of the `INITIALIZE/QUEUE` command. You can supply the information in a "configuration" system logical name that the symbiont checks. You can use both, putting some information on one place, and some in the other. You can also put configuration information in one or more files and reference those files from the `/DESCRIPTION` string and/or "configuration" logical name (see the `/INCLUDE=` option), or even from other such files. If you have large numbers of queues making up complicated groupings with similar requirements, this flexibility can help reduce the time required to set up and maintain queues.

In addition to the basic configuration information, there are several optional logical names used to control specific behaviors. Note that the default behaviors may be adequate to your needs.

The following sections describe the various logical names, queue settings, and `PRINT` command options available with the IPP symbiont. In many cases there is a "Global" version, which affects all IPP symbiont queues on the system, as well as a "queue-specific" version that affects only a specified queue. `PRINT` command options affect only the job being submitted.

# Global Settings

These logical names establish configuration values for all queues on the system, not on a queue-by-queue basis. Where there are queue-specific settings related to these, these become the default values, overriding any built-in defaults.

## **MULTINET\_IPP\_CONFIG**

Specifies one or more of the qualifiers described in the *Queue-specific Settings* section. These qualifiers are not case sensitive. Underscores ( ) in the qualifier names are optional. Each may be abbreviated as long as the result is not ambiguous. There is no default. This logical provides defaults that may be overridden by the queue-specific configuration logical, `MULTINET_IPP_queue_name_CONFIG`, for a given queue.

## **MULTINET\_IPP\_DEFAULT\_DOCUMENT\_FORMAT**

Specifies a string to use as the document format, unless specified differently for a given queue or print job. The actual document format used on a given job must be a valid MIME media type, supported by the printer to which the job is sent. The default is "text/plain".

## **MULTINET\_IPP\_DOCALIAS\_FILE**

Specifies document format name aliasing. Rather than having to specify long mime-media-type names for document formats, you can define local names that are equivalent, and the symbiont will do the replacement. For example, you can define "PS" as equivalent to "application/postscript", and use it in print commands as `/DOCUMENT_FORMAT=PS`. There is an escape mechanism in case a logical name is ever made into a different MIME-media-type. Prefixing the document format name with % prevents alias translation. `%PS` means just send it as PS, do not translate PS into `APPLICATION/POSTSCRIPT` in the request.

To use aliasing, define the system logical name `MULTINET_IPP_DOCALIAS_FILE` with the filename of the alias file as the equivalence string. The format of the alias file is:

```
MULTINET_IPP_mime-name
```

Blank lines are ignored. Lines starting with # are treated as comments and are ignored. Document format name aliasing has been added. Rather than having to specify long mime-media-type names for document formats, you can now define local names that are equivalent, and the symbiont will do the replacement. For example, you can define "PS" as equivalent to "application/postscript", and use it in print commands such as `/DOCUMENT_FORMAT=PS`.

There is an "escape mechanism" in case a local name is ever made into a different MIME-media-type. Prefixing the document format name with "%" prevents alias translation. "%PS" means "just send it as 'PS', don't translate "PS" into "application/postscript" in the request.

To use aliasing, define the system logical name `MULTINET_IPP_DOCALIAS_FILE` with the filename of the alias file as the equivalence string.

The format of lines in the alias file is:

```
Mime-name: alias, alias, alias...
```

Blank lines are ignored, and lines starting with "#" are treated as comments and are ignored.

#### **MULTINET\_IPP\_IGNORE\_DESCRIPTION**

#### **MULTINET\_IPP\_queue\_name\_IGNORE\_DESCRIPTION**

If this logical is defined, the symbiont ignores the /DESCRIPTION strings for all IPP queues. This allows use of /DESCRIPTION for other information without affecting the symbiont. Configuration of the symbiont must be done through use of the `MULTINET_IPP_CONFIG` logical, or the queue-specific logical, `MULTINET_IPP_queue_name_CONFIG` if `MULTINET_IPP_IGNORE_DESCRIPTION` is defined. The value of the equivalence string for `MULTINET_IPP_IGNORE_DESCRIPTION` is not important. The existence or non-existence of the logical is all that is checked. This logical provides defaults that may be overridden by the queue-specific configuration logical, `MULTINET_IPP_queue_name_IGNORE_DESCRIPTION`, for a given queue.

#### **MULTINET\_IPP\_JOB\_RETRY\_DELAY**

Specifies, as an OpenVMS delta time specification, the length of time to hold a job when it is re-queued due to a temporary problem. The default value is "0 00:10:00.00" (10 minutes).

#### **MULTINET\_IPP\_MAX\_LOG\_BYTES**

Specifies how many bytes of data will be logged by the send and receive routines when running with logging level set to `DETAILED_TRACE`. The value is an integer. A negative value sets the limit to infinite (all data will be logged). A value of zero turns off inclusion of data to the log file. A positive value sets the actual number of bytes logged, and any additional data is ignored. The default action is to log all data.

#### **MULTINET\_IPP\_MAX\_STREAMS**

Specifies the number of streams (queues) that each IPP symbiont process can handle. This is an integer from 1 to 16. The default is 16.

#### **MULTINET\_IPP\_LOG\_LEVEL**

Specifies one of the logging level values listed in the logging level table below, and is used to determine how serious a message must be before it is written to the log file. Only those messages marked as this level, or as a more serious level, are logged. The default is `JOB_TRACE`.

#### **MULTINET\_IPP\_LOGFILE**

Specifies the name of the log file. All queues for a given symbiont process will share this file unless there are individual queue overrides. The default is to create the log file in the default spool directory, with the name `IPP_SYMBIONT_pid.LOG`.

#### **MULTINET\_IPP\_OPCOM\_LEVEL**

Specifies one of the logging level values listed in the logging level table below, and is used to determine how serious a message must be before it is sent to OPCOM. Only those messages marked as this level, or as a more serious level, are sent. The default is `INFO`.

#### **MULTINET\_IPP\_OPCOM\_TERMINAL**

Specifies the OPCOM operator "terminal" to send OPCOM messages to. Permissible values are listed later in this section. The default is the `PRINT` operator.

## **Queue-specific Settings**

These items are specified as qualifiers in the queue's `/DESCRIPTION` string, and/or in the `MULTINET_IPP_queue_name_CONFIG` logical equivalence string - the two are concatenated before being processed. These qualifiers are not case sensitive. The underscores in the qualifier names are optional. Each may be abbreviated as long as the result is not ambiguous. The two sections below contain the complete list of qualifiers.

### **Queue-specific Required Qualifier**

#### **`/PRINTER_URI`**

A valid URI, or list of URIs, for the printer or printers to be sent to from this queue. Wildcards are allowed ("`*`" to match one or more characters, "`?`" for a single character). The individual URIs in the list are separated from each other with the vertical bar ("`|`") character. The first URI in the list that does not include any wildcards is the default printer for the queue. If there are no default printer URIs and you have not specified a particular printer URI with the `PRINT` command, the job is aborted. Any printer URI specified with the `PRINT` command must match at least one of the URIs listed for the queue or the job will be aborted.

## Queue-specific Optional Qualifiers

### ***/COMMENT=quoted string***

Allows inclusion of a quoted string of text that the symbiont will ignore, other than to write to the log file and/or OPCOM if the logging level is set to SYMBIONT or a more detailed setting.

### ***/COPIES\_DEFAULT=number***

Specifies the number of copies of each document to print unless specified otherwise on the PRINT command. The default value is 1.

### ***/DEBUG***

Causes the symbiont to retain all spool files and to force DETAILED\_TRACE logging to the log file, regardless of what other settings might be specified. Note that /DEBUG forces the setting for MAX\_LOG\_BYTES to a minimum of 512 bytes. You can set it higher, but any setting lower than 512 bytes will be ignored when /DEBUG is used.

### ***/DEFAULT\_DOCUMENT\_FORMAT=formatspec or /DOCUMENT\_FORMAT\_DEFAULT=formatspec***

Specifies the default document format for the queue. This value will be a MIME media type that is supported for the printer or printers served by this queue. It could also be the string "\*\*\*printer\_default\*\*\*", which will result in whatever the target printer defines as its default when no document format is specified on the PRINT command.

### ***/EXPECT\_LINK\_CLOSURES***

Specifies that the printer is not fully HTTP 1.1 compliant because it does not support persistent connections, and does not send a "Connection: Close" header line in its last response. Therefore, the symbiont should assume that such a line was sent in every response, using a new link for each request, closing the old one, and not treating it as an error if the other end closes the link after sending a response.

### ***/FINISHINGS\_DEFAULT=keyword***

Specifies finishing operations to be performed on the printed documents. May or may not be supported by a given IPP server. Any or all of the four available finishings may be specified. Case is ignored.

Keywords are:

NONE

STAPLE  
PUNCH  
COVER  
BIND

***/[NO]FLAG\_DEFAULT***

Specifies whether a `job-sheets` attribute will be specified for jobs by default. If

`/FLAG_DEFAULT` is used, `job-sheets` will be requested as “standard”. If `/NOFLAG_DEFAULT` is used, `job-sheets` will be requested as “none”.

***/INCLUDE=filename***

Specifies a sequential access text file containing additional qualifiers from this list. These qualifiers are read and processed at the point where the `/INCLUDE` qualifier is encountered, and share the precedence of that point.

***/JOB\_PRIORITY\_DEFAULT=integer***

Specifies the priority of the print job. 1 is the lowest, 100 is the highest.

***/JOB\_RETRY\_DELAY=deltatime***

Specifies, as an OpenVMS delta time specification, the length of time to hold a job when it is re-queued due to a temporary problem. The default value is "0 00:10:00.00" (10 minutes).

***/LOG\_FILE=filename***

Specifies the name of the queue log file to write messages to for this queue. The default is in MultiNet's default spool directory, unless overridden by a global setting, as described in *Global Settings*. The default filename is `IPP_SYMBIONT_Process_PID.LOG`.

***/LOG\_LEVEL=logging\_level***

Specifies one of the Logging Levels values listed in the table below, and is used to determine the severity of a message before it is written to the queue log file. Only those messages marked as this level, or a more serious one, are logged. The default is `JOB_TRACE` unless overridden by a global `MULTINET_IPP_LOG_LEVEL` logical.

***/MAX\_LOG\_BYTES=number***

Specifies how many bytes of data will be logged by the send and receive routines when running with logging level set to `DETAILED_TRACE`. The value is an integer. A negative value sets the limit to infinite (all data will be logged). A value of zero turns off inclusion of data to the log file. A positive value sets the actual number of bytes logged, and any additional data is ignored. The default action is to log all data.

**`/MEDIA_DEFAULT=name`**

This attribute identifies the medium that the printer uses for all pages of the Job.

The values for "media" include medium-names, medium-sizes, input-trays and electronic forms. See your printer documentation for details concerning what values are supported for your printer.

Standard keyword values are taken from ISO DPA and the Printer MIB and are listed in Section 14 of RFC 2566. Some servers may support definition of locally created names as well. See Table 16-1 for standard values for input trays. The below table contains examples of standard names. These names include, but are not limited to the following:

<b>Name</b>	<b>Description</b>
<code>default</code>	The default medium for the output device
<code>iso-a4-white</code>	Specifies the ISO A4 white medium
<code>iso-a4-colored</code>	Specifies the ISO A4 colored medium
<code>iso-a4-transparent</code>	Specifies the ISO A4 transparent medium
<code>na-letter-white</code>	Specifies the North American letter white medium
<code>na-letter-colored</code>	Specifies the North American letter colored medium
<code>na-letter-transparent</code>	Specifies the North American letter transparent medium
<code>na-legal-white</code>	Specifies the North American legal white medium
<code>na-legal-colored</code>	Specifies the North American legal colored medium
<code>na-9x12-envelope</code>	Specifies the North American 9x12 envelope medium



monarch-envelope	Specifies the Monarch envelope
na-number-10-envelope	Specifies the North American number 10 business envelope medium
na-7x9-envelope	Specifies the North American 7x9 inch envelope
na-9x11-envelope	Specifies the North American 9x11 inch envelope
na-10x14-envelope	Specifies the North American 10x14 inch envelope
na-number-9-envelope	Specifies the North American number 9 business envelope
na-6x9-envelope	Specifies the North American 6x9 inch envelope
na-10x15-envelope	Specifies the North American 10x15 inch envelope
executive-white	Specifies the white executive medium
folio-white	Specifies the folio white medium
invoice-white	Specifies the white invoice medium
ledger-white	Specifies the white ledger medium
quarto-white	Specifies the white quarto medium
iso-a0-white	Specifies the ISO A0 white medium
iso-a1-white	Specifies the ISO A1 white medium
a	Specifies the engineering A size medium
b	Specifies the engineering B size medium
c	Specifies the engineering C size medium
d	Specifies the engineering D size medium

e	Specifies the engineering E size medium
---	---

The following standard values are defined for input-trays:

Name	Description
top	The top input tray in the printer.
middle	The middle input tray in the printer.
bottom	The bottom input tray in the printer.
envelope	The envelope input tray in the printer.
manual	The manual feed input tray in the printer.
large-capacity	The large capacity input tray in the printer.
main	The main input tray
side	The side input tray

**/MULTIPLE\_DOCUMENT\_HANDLING\_DEFAULT=keyword**

This qualifier is relevant only for jobs consisting of two or more documents, and when the IPP server supports jobs with multiple documents. The qualifier controls finishing operations and the placement of one or more pages onto media sheets. When printing multiple copies, it also controls the order in which the copies that result are produced. Standard keyword values are:

**single-document**

If a job has multiple documents, say, the documents are called A and B, then the result printing the data (A and then B) will be treated as a single sequence of media sheets for finishing operations; that is, finishing would be performed on the concatenation of the two documents. The printer will not force the data in each document to start on a new page.

If more than one copy is requested, the ordering of the pages resulting from printing will be A, B, A, B, ..., and the printer will force each copy (A, B) to start on a new media sheet.

### **separate-documents-uncollated-copies**

If a job has multiple documents, say, the documents are called A and B, then the result of printing each document will be treated as a single sequence of media sheets for finishing operations; that is, the documents A and B would each be finished separately. The printer will force each copy of the data in a single document to start on a new sheet.

If more than one copy is made, the ordering of the pages will be A, A, ..., B, B ... .

### **separate-documents-collated-copies**

If a job has multiple documents, say, A and B, then the result will be that each document will be treated as a single sequence of media sheets for finishing operations; that is, A and B would each be finished separately. The printer will force each copy of the result of processing the data in a single document to start on a new sheet.

If more than one copy is made, the ordering of the documents will be A, B, A, B,... .

### **single-document-new-sheet**

Same as `single-document`, except that the printer will ensure that the first page of each document in the job is placed on a new media sheet.

The `single-document` value is the same as `separate-documents-collated-copies` with respect to ordering of print-stream pages, but not media sheet generation, since `single-document` will put the first page of the next document on the back side of a sheet if an odd number of pages have been produced so far for the job, while `separate-documents-collated-copies` always forces the next document or document copy on to a new sheet. In addition, if the `finishings` attribute specifies `staple`, then with `single-document`, documents A and B are stapled together as a single document with no regard to new sheets, with `single-document-new-sheet`, documents A and B are stapled together as a single document, but document B starts on a new sheet, but with `separate-documents-uncollated-copies` and `separate-documents-collated-copies`, documents A and B are stapled separately.

**Note:** None of these values provide means to produce uncollated sheets within a document, i.e., where multiple copies of sheet n are produced before sheet n+1 of the same document.

***/NUMBER\_UP\_DEFAULT=number***

Specifies the number of page images to be placed on each side of each sheet of paper. The number must be an integer that is acceptable to the IPP server. If the number specified is not a value supported by the server, the job aborts.

***/OPCOM\_LEVEL=logging\_level***

Specifies one of the logging level values listed in the table below, and is used to determine the severity of a message before it is sent to OPCOM. Only those messages marked as this level, or at a more serious level, are sent. The default is INFO unless overridden by a global MULTINET\_IPP\_OPCOM\_LEVEL logical.

***/OPCOM\_TERMINAL=opcom\_term***

Specifies which OPCOM operator "terminal" to send OPCOM messages to. Available values are listed in the OPCOM Terminal Names table below. The default is the PRINT operator. See the OpenVMS documentation for the REPLY/ENABLE command for more information on OPCOM terminals.

***/ORIENTATION\_DEFAULT=keyword***

Specifies the default page orientation. Case is ignored. Supported values are:

PORTRAIT  
REVERSE\_PORTRAIT  
LANDSCAPE  
REVERSE\_LANDSCAPE

***/PAGE\_RANGE\_DEFAULT="range[,range]..."***

Specifies the page numbers to print. range is either a single integer page number, or a pair of page numbers, separated by a hyphen. Multiple range specifications are separated by commas. For example:

```
$ PRINT/QUEUE=IPP_QUEUE/PARAM=(PAGE_RANGES="1,3-6,9,10,12-14") TEST.TXT
```

The example specifies the pages: 1, 3, 4, 5, 6, 9, 10, 12, 13, and 14. Note that embedded spaces are allowed, and ignored.

***/QUALITY\_DEFAULT=keyword***

Specifies the quality of the printed material. Case is ignored. The keywords are:

DRAFT  
NORMAL  
HIGH

***/SIDES\_DEFAULT=keyword***

Specifies how the printing is to be placed on the paper.

- ONE-SIDED: prints each consecutive page upon one side of consecutive media sheets.
- TWO-SIDED-LONG-EDGE: prints each consecutive pair of pages upon the front and back sides of consecutive media sheets, with the orientation of each pair of pages on the long edge. This positioning is called “duplex” or “head-to-head” also.
- TWO-SIDED-SHORT-EDGE: prints each consecutive pair of pages upon front and back sides of consecutive media sheets, with the orientation of each pair of print-stream pages on the short edge. This positioning is called “tumble” or “head-to-toe” also.

***/SPOOL\_DIRECTORY=dirspec***

Specifies the directory to use for storing temporary files used while processing print jobs for the queue. The default is MultiNet's default spool directory.

## Order of Processing

The various logicals and qualifiers described in the previous two sections sometimes define the same configuration item. The operation has been defined, but the precedence has not. The order, from lowest precedence to highest, is:

1. Built-in hard coded default values.
2. Global logicals, as described in the first section.
3. Queue-specific qualifiers found in the */DESCRIPTION* string of the queue.
4. Queue-specific qualifiers found in the queue-specific *CONFIG* logical.

The queue-specific qualifiers are parsed second, allowing for an override of the global settings on a queue-by-queue basis when that behavior is desired.

## Print Command Options

Print command options are specified using the OpenVMS standard */PARAMETERS* qualifier. The list of options is enclosed in parenthesis. For example,

```
$ PRINT /QUEUE=IPP PRINTER_1 /PARAMETER=(COPIES=3, ORIENTATION=LANDSCAPE)  
FILE.TXT
```

These options are not case sensitive. The underscores in the option names are optional. Each may be abbreviated as long as the result is not ambiguous. The available print command options are:

**PRINTER=*printer\_uri***

Specifies the target printer when the queue default is not desired, or when there is no queue default. The printer URI specified must match at least one of the defined *printer\_uri*'s for the print queue.

Wildcards cannot be used in the printer URI.

**COPIES=*number***

Specifies the number of copies of each document to print. The default value is 1.

**SIDES=*keyword***

Specifies how the printing is to be placed on the paper. The *keyword* must be one of the following:

- ONE-SIDED or *1sided*: prints each consecutive page upon one side of consecutive media sheets.
- TWO-SIDED-LONG-EDGE or *two-long-edge* or *2long\_side*: prints each consecutive pair of pages upon the front and back sides of consecutive media sheets, with the orientation of each pair of pages on the long edge. This positioning is called “duplex” or “head-to-head” also.
- TWO-SIDED-SHORT-EDGE or *two-short-edge* or *2short\_side*: prints each consecutive pair of pages upon front and back sides of consecutive media sheets, with the orientation of each pair of print-stream pages on the short edge. This positioning is called “tumble” or “head-to-toe” also.

**ORIENTATION=*keyword***

Specifies the page orientation. The *keyword* must be one of:

```
PORTRAIT  
REVERSE_PORTRAIT  
LANDSCAPE  
REVERSE_LANDSCAPE
```

These can be abbreviated to any non-ambiguous prefix. Case is ignored.

**[NO] FLAG**

Requests, or suppresses, the printing of an IPP flag page for the job. The printer may, or may not, respond to this request. The exact format of this flag page is up to the IPP Server (printer) implementation.

**NUMBER\_UP=*number***

Specifies the number of page images to be placed on each side of each sheet of paper. The number must be an integer that is acceptable to the IPP server. If the number specified is not a value supported by the server, the job aborts.

**DOCUMENT\_FORMAT=*MIME-media-type***

**DOCUMENT\_FORMAT=\*\*\**printer\_default*\*\*\***

Specifies the document format of the files in the job, or specifies use of the printer's built-in default. The default for this qualifier is the default for the queue. Also, if the queue configuration does not specify a default document format, the hard coded default is "text/plain".

**JOB\_PRIORITY=*integer***

Specifies the priority of the print job at the IPP server (not to be confused with the OpenVMS queue priority). 1 is the lowest, 100 is the highest.

**FINISHINGS="*keyword[,keyword]...*"**

Specifies finishing operations to be performed on the printed documents. May or may not be supported by a given IPP server. Any or all of the four available finishings may be specified. Case is ignored.

BIND  
COVER  
PUNCH  
STAPLE

**MULTIPLE\_DOCUMENT\_HANDLING=*keyword***

Specifies how you want the printer to print your job. The keyword is one of the following:

- `Single_Document` or `1Document`
- `Separate_Documents_Uncollated_Copies` or `UncollatedSeparate`
- `Separate_Documents_Collated_Copies` or `CollatedSeparate`
- `Single_Document_New_Sheet` or `NewSheet`

Case is ignored. See */MULTIPLE\_DOCUMENT\_HANDLING\_DEFAULT* in this chapter for information on single document, separate-documents-uncollated-copies, separate-documents-collated-copies, and single-document-new-sheet handling.

**PAGE\_RANGES=" *range* [, *range*] . . . "**

Specifies the page numbers to print. *range* is either a single integer page number, or a pair of page numbers, separated by a hyphen. Multiple range specifications are separated by commas and enclosed in double quotes.

For example:

```
$ PRINT/QUEUE=IPP QUEUE/PARAM=(PAGE_RANGES="1,3-6, 9, 10, 12-14") FILE.TXT
```

Note that embedded spaces are allowed, and ignored. The example specifies the pages: 1, 3, 4, 5, 6, 9, 10, 12, 13, and 14.

**MEDIA=*name***

This attribute identifies the medium that the Printer uses for all pages of the Job.

The values for "media" include medium-names, medium-sizes, input-trays and electronic forms. See your printer documentation for details concerning what values are supported for your printer.

Standard keyword values are taken from ISO DPA and the Printer MIB and are listed in section 14 of RFC 2566. Some servers may support definition of locally created names as well.

See the tables earlier in this chapter for the standard media names.

**QUALITY=*keyword***

Specifies the quality of the printed material. Case is ignored. The keyword choices are:

DRAFT  
HIGH  
NORMAL

## Allowable Values

Several of the configuration and job submission settings require values for OPCOM terminal names or logging severity levels. This section defines the allowable values for these options.



# OPCOM Terminal Names

CARDS	NETWORK	OPER3	OPER9
CENTRAL	PRINTER (default)	OPER4	OPER10
CLUSTER	SECURITY	OPER5	OPER11
DEVICES	TAPES	OPER6	OPER12
DISKS	OPER1	OPER7	NONE (do NOT send to OPCOM...
LICENSE	OPER2	OPER8	except OVERRIDE events)

# Logging Levels

All values may be abbreviated to any non-ambiguous prefix. These values are not case sensitive.

DETAILED_TRACE	All events
FILE	Events related to processing of individual files
JOB	Events related to processing of individual jobs
SYMBIONT	Events related to the state of the symbiont
INFO	Events providing information about non-error conditions
WARNING	Events warning of potential problems that do not halt processing
ERROR	Events reporting an error that prevented processing of a job
FATAL	Events reporting an error that stopped a queue
ABORT	Events reporting an error that caused the symbiont to exit

There are a few messages that are marked to be reported regardless of the setting of the various OPCOM and log file severity levels. These are kept to a minimum, but are considered to be important enough to override the logging level settings. These cannot be suppressed.

# Using Logicals to Define Queue Configurations

This section provides examples of using logicals to define queue configuration prior to queue initialization. This method can be used both as an alternative to and in addition to the `/DESCRIPTION` string shown in the previous examples. See the *Configuration* section for a complete description of all available qualifiers.

## Setting Up IPP Symbiont Queues

Creating an IPP symbiont queue is done using the OpenVMS `INITIALIZE/QUEUE` command. All standard qualifiers are allowed, but the `/DESCRIPTION` qualifier has special use with the IPP symbiont. See the *Configuration* section.

## Setting up IPP Symbiont Queues Using Queue-Specific Logicals

Set up an IPP symbiont queue named `ENG_PRINTER` to obtain its configuration information from a queue specific configuration file and to print a flag page with each job.

```
$ DEFINE/SYSTEM MULTINET_IPP_ENG_PRINTER_CONFIG -
_ $ "/INCLUDE=SYS$SYSTEM:ENG_PRINTER.SETUP/FLAG_DEFAULT"
$ INITIALIZE/QUEUE/PROCESSOR=MULTINET_IPP_SYMB ENG_PRINTER
```

The file `SYS$SYSTEM:ENG_PRINTER.SETUP` contains:

```
/printer="ipp://engprinter.example.com:631/ipp"
```

# Setting Up an IPP Symbiont Queue to Print Only to a Specific Printer

Set up the IPP symbiont queues named `IPP_PRINT_QUEUE` and `IPP_PRINT_2` to print only to the `iprinter.example.com` printer on port 631. Additionally, `IPP_PRINT_2` will always print two copies of each submitted file if copies are supported by the printer.

```
$ DEFINE/SYSTEM MULTINET IPP_CONFIG -
_$ "/PRINTER_URI="ipp://iprinter.example.com:631/ipp""
_$ INITIALIZE/QUEUE /PROCESSOR=MULTINET_IPP_SYMB IPP_PRINT_QUEUE
_$ INITIALIZE/QUEUE /PROCESSOR=MULTINET_IPP_SYMB -
_$ /DESCRIPTION="/copies_default=2" IPP_PRINT_2
```

# Setting Up to Print to Multiple Printers Using Wildcards

Set up an IPP symbiont queue to print to any IPP printer in the `example.com` domain, with the default printer being `iprinter.example.com`:

```
$ INITIALIZE/QUEUE /PROCESSOR=MULTINET_IPP_SYMB /DESCRIPTION="/printer=
"http://iprinter.example.com:631/ipp|*.example.com"" IPP_PRINT_QUEUE
```

# Setting Up Two Queues Using a Disk File for Queue Settings

Set up two IPP symbiont queues to print to any printer in the `example.com` domain, with the default printer being `iprinter.example.com` for one queue, and `oprinter.example.com` for the other. Log all possible messages to the log file, but send only messages more severe than `FILE_TRACE` to `OPCOM`. Use a 5 minute retry delay, and make the document format default the same as the printer's default. Use a disk file for the configuration information common to both queues:

```
$ INITIALIZE/QUEUE /PROCESSOR=MULTINET_IPP_SYMB -
_$ /DESCRIPTION="/printer=
"http://iprinter.example.com:631/ipp|*.example.com""
/include=SYS$SYSTEM:IPP_QUEUE.SETUP" IPRINTER_QUEUE
```

```
$ INITIALIZE/QUEUE /PROCESSOR=MULTINET_IPP_SYMB -
_$ /DESCRIPTION="/printer=
"http://oprinter.example.com:631/ipp|*.example.com""
/include=SYS$SYSTEM:IPP_QUEUE.SETUP" OPRINTER_QUEUE
```

The file `SYS$SYSTEM:IPP_QUEUE.SETUP` contains:

```
/log_level=DETAILED_TRACE
/opcom_level=FILE_TRACE
/job_retry_delay="0 00:05:00.00"
/default_document_format=***printer_default***
```

## Setting Up Two Queues with no Configuration Values in the INITIALIZE Command

Do the same as the prior example, but put as much of the configurations in disk files as possible to allow changes to queue characteristics without having to re-initialize the queues:

```
$ INITIALIZE/QUEUE /PROCESSOR=MULTINET_IPP_SYMB -
_$ /DESCRIPTION="/INCLUDE=SYS$SYSTEM:IPP_IPRINTER.SETUP" IPRINTER_QUEUE
```

```
$ INITIALIZE/QUEUE /PROCESSOR=MULTINET_IPP_SYMB -
_$ /DESCRIPTION="/INCLUDE=SYS$SYSTEM:IPP_OPRINTER.SETUP" OPRINTER_QUEUE
```

The file SYS\$SYSTEM:IPP\_IPRINTER.SETUP contains:

```
/printer="http://iprprinter.example.com:631/ipp|*.example.com"
/include=SYS$SYSTEM:IPP_QUEUE.SETUP
```

The file SYS\$SYSTEM:IPP\_OPRINTER.SETUP contains:

```
/printer="http://oprprinter.example.com:631/ipp|*.example.com"
/include=SYS$SYSTEM:IPP_QUEUE.SETUP
```

The file SYS\$SYSTEM:IPP\_QUEUE.SETUP contains:

```
/log_level=DETAILED_TRACE
/opcom_level=FILE_TRACE
/job_retry_delay="0 00:05:00.00"
/default_document_format=***printer_default***
```

## Submitting Jobs to IPP Symbiont Print Queues

This section describes how to submit jobs to the IPP symbiont print queues.

## Printing a Single Text File to an IPP Queue

Print the file `FOO.TXT` to the `IPRINTER` (default destination printer) set up in the prior examples:

```
$ PRINT/QUEUE=IPRINTER_QUEUE foo.txt
```

## Specifying the Destination Printer on the Print Command

Print a single text file to a non-default printer on a queue with a wild carded printer URL:

```
$ PRINT /QUEUE=iprinter_queue -  
$ /PARAM=(printer="ipp://another.example.com/ipp/port1") foo.txt
```

**Note:** The above will fail unless the queue specifies `another.example.com` as a legal URL, either explicitly or by using wildcards.

## Using Other Print Qualifiers

Print a text file to a default printer on a queue but specify the document format and additional copies:

```
$ PRINT /QUEUE=iprinter_queue /PARAM=(document="plain/text",copies=3)  
foo.txt
```

## MULTINET IPP SHOW Command

The `MULTINET IPP SHOW` utility allows a user to learn the capabilities supported by an IPP server.

The command syntax is:

```
$ MULTINET IPP SHOW server_URI /qualifiers...
```

Refer to the MultiNet Administrator's Reference Guide, Chapter 1, for details.

# 14. RMT Server and Client Configuration

This chapter explains how to configure the RMT (Remote Magnetic Tape) server, and how to use RMTALLOC (the RMT client) with tape drives and CD-ROM drives.

## Configuring the Remote Magnetic Tape Server

The MultiNet remote magnetic tape server (RMT) uses the BSD RMT protocol to allow UNIX and PC users to access tape drives on OpenVMS systems. Most UNIX-type systems support the `rdump` and `rrestore` commands for accessing tape drives served by RMT.

To enable and configure the MultiNet RMT server:

1. Make sure RSHELL works from the UNIX Operating system user `root` to the OpenVMS user `ROOT`. If no OpenVMS user `ROOT` exists, the RMT server uses the OpenVMS user `SYSTEM`.

**Note:** For `ROOT` and `SYSTEM`, the system-wide `MULTINET:HOSTS.EQUIV` file is ignored and an explicit entry in `SYS$LOGIN:.RHOSTS` is required to grant access.

2. Make sure the `ROOT/SYSTEM LOGIN.COM` and the system-wide `SYLOGIN.COM` do not print anything when you issue remote `RSHELL` (under OpenVMS) or `rsh` (under the UNIX Operating System or on PCs) commands. Anything written to `SYS$OUTPUT` from these command procedures interferes with the RMT protocol.

The following example shows commands that prevent output from being displayed by `SYSTEM/ROOT LOGIN.COM` and `SYLOGIN.COM`.

```
$ VERIFY = 'F$VERIFY(0) ! Turn off verify without echoing
```

```

$ IF F$MODE() .EQS. "OTHER" THEN EXIT ! If a DETACHED process (RSHELL) .
.
$ IF VERIFY THEN SET VERIFY ! If a batch job, may want to turn
! verify back on.

```

You can specify either UNIX- or OpenVMS-style magtape device names or an OpenVMS file name for writing to a disk file.

When you specify UNIX-style names, options are encoded in the unit number (minor device number). The correspondence between the options and their associated unit numbers is as follows:

Device	Options				
mt0 through mt3	/NOMOUNT	/STREAM	/DENS=800	/REWIND	/NOUNLOAD
mt4 through mt7	/NOMOUNT	/STREAM	/DENS=800	/NOREWIND	/NOUNLOAD
mt8 through mt11	/NOMOUNT	/STREAM	/DENS=160	/REWIND	/NOUNLOAD
mt12 through mt15	/NOMOUNT	/STREAM	/DENS=160	/NOREWIND	/NOUNLOAD
mt16 through mt19	/NOMOUNT	/STREAM	/DENS=625	/REWIND	/NOUNLOAD
mt20 through mt23	/NOMOUNT	/STREAM	/DENS=625	/NOREWIND	/NOUNLOAD
rmt0 through rmt3	/NOMOUNT	/NOSTREAM	/DENS=800	/REWIND	/NOUNLOAD
rmt4 through rmt7	/NOMOUNT	/NOSTREAM	/DENS=800	/NOREWIND	/NOUNLOAD
rmt8 through rmt11	/NOMOUNT	/NOSTREAM	/DENS=16	/REWIND	/NOUNLOAD
rmt12 through rmt15	/NOMOUNT	/NOSTREAM	/DENS=16	/NOREWIND	/NOUNLOAD
rmt16 through rmt19	/NOMOUNT	/NOSTREAM	/DENS=62	/REWIND	/NOUNLOAD
rmt20 through rmt23	/NOMOUNT	/NOSTREAM	/DENS=62	/NOREWIND	/NOUNLOAD

The OpenVMS tape drive name is chosen automatically as the first tape drive, or you can set it using the `NET-CONFIG SET DEFAULT-RMT-TAPE-DEVICE` command.

When you specify OpenVMS-style names, the options are encoded in qualifiers; the exact format is:  
 vms\_node\_name:volume\_name[/qualifiers[...]]

For example:

```
# rdump 0f example.com:/dev/rmt8 /usr
```

or:

```
# rdump 0f example.com:mua0:/nomount/nostream /dens=1600/nounload /USR
```

or:

```
# rdump 0f abc.com:mua0:xxx/nostream /dens=1600/nounload \
/comment="Please mount volume XXX on drive mua0" /usr
```

The table below lists the qualifiers available for OpenVMS tape drive names.

Qualifiers	Description
/BLOCKSIZE= <i>size</i>	Block size at which to write the tape. Default: 65534 bytes.
/DENSITY= <i>density</i>	Specifies the density at which to write a tape. Default: current density.
/[NO]REWIND	Specifies whether to rewind the drive on close; ignored unless /NOMOUNT is specified. Default: /REWIND.
/[NO]UNLOAD	Specifies whether to unload the drive on close. Default: /UNLOAD.
/COMMENT=" <i>string</i> "	Comment to display in the remote OPCOM message, either appended to or replacing the default text, depending on the resulting string length being less than the 78-character maximum. This message is the only opportunity to send a tape-specific message to the remote operator. (MOUNT/COMMENT strings are not passed to a remote system.) Because RMTALLOC will not complete until a tape has been loaded and the drive is online, use COMMENT to make sure the operator is aware of your request.
/[NO]MOUNT	Mounts the tape drive using the OpenVMS MOUNT service.  /NOMOUNT accesses the tape drive without mounting it. This qualifier is used for UNIX utilities which expect the tape drive to hold its current position (not



	rewind) if they close it. By not mounting it, the tape drive does not rewind when dismounted. Default: /MOUNT.
/ [NO] STREAMING	Accesses the tape drive as a sequential device (a UNIX character device). /NOSTREAMING accesses the tape drive as a raw device (a UNIX block device). Default: /STREAMING.

## About the RMT Client

The RMT client `MULTINET RMTALLOC` is used for accessing tape or CD-ROM drives on remote hosts over TCP (using `RSHELL`). If restrictions apply where `RSHELL` does not work, or if `RSHELL` outputs spurious login messages or greetings, `RMTALLOC` does not work. `RMTALLOC` depends on an RMT server to function properly. `RMTALLOC` creates a pseudo device that appears as an OpenVMS physical device to the OpenVMS `BACKUP`, `COPY`, and other utilities. The pseudo device is named `RMTx:`, `x` is the unit number. The actual tape or CD-ROM drive can be on another MultiNet OpenVMS system or on any host running the RMT server, such as those running the BSD or SunOS UNIX operating system.

For CD-ROM, `RMTALLOC` treats the drive as a file system, which speeds file access.

There are some limits to the types of tape devices you can access on other operating systems and the amount of control available. Because UNIX tapes and tape drivers cannot write variable-length blocks and do not allow skipping forward over records between read operations, they cannot be used with OpenVMS `BACKUP` or `COPY` commands.

## Limitations of UNIX Devices and Software

The following list describes known limitations of common UNIX devices and software:

- UNIX QIC tape drives cannot be used.
- Solaris RMT servers require you to use `/UNIX_SERVER=BROKEN` so you can back up one single-volume `BACKUP` saveset. Use `BACKUP/REWIND` to copy to or from tape.
- ULTRIX/Tru64 RMT servers require you to use `/UNIX_SERVER=ULTRIX` to obtain full OpenVMS tape functionality.
- SGI's IRX RMT server (as of 4.0.5) does not interoperate with OpenVMS tape operations.
- IBM RS6000/AIX requires a special translation on the IBM side because IBM uses incompatible RMT commands.

# Using RMTALLOC

To use RMTALLOC:

1. Make sure the media is in the drive and the drive is online. If you verify this, use the `/SEMANTICS=(COMMENT=comment)` qualifier to inform the remote VMS operator of a pending device mount.
2. Allocate the drive with RMTALLOC.
3. Mount the tape or CD-ROM drive. Use the same MOUNT command you would for a VMS device.
4. Read from or write to the tape, or read the information from the CD-ROM.
5. Dismount the tape or CD-ROM drive.
6. Deallocate the tape or CD-ROM drive.
7. Remove the media from the drive.

In its simplest form, you can specify RMTALLOC as follows:

```
$ MULTINET RMTALLOC hostname::devicename
```

For example, for a tape device, enter:

```
$ RMTALLOC SFO.EXAMPLE.COM::MUA0: MYTAPE
```

For a CD-ROM drive, enter:

```
$ RMTALLOC /CD CONTROL::DISK$CD: MYCD /USER=SYSTEM
```

These examples are explained further in the following sections.

## RMTALLOC Tape Drive Access Example

The following example shows how to allocate a tape drive and write to tape using the OpenVMS TAR utility:

```
$ RMTALLOC BOS.EXAMPLE.COM::MUA0: MYTAPE
%RMT-I-ALLOC, _MYSYS$RMT1: allocated (BOS.EXAMPLE.COM::MUA0:)

$ MOUNT /FOREIGN /RECORD SIZE=512 /BLOCK SIZE=10240 MYTAPE
%MOUNT-I-MOUNTED, MYTAPE mounted on _MYSYS$RMT1:

$ TAR /ARCHIVE=MYTAPE WRITE AFILE.TXT
```

```
%TAR-S-WRITTEN, written USERS:[ME]AFILE.TXT;1 (13495 bytes)
%TAR-S-TOTWRITE, total of 1 file written
```

```
$ DISMOUNT _MYSYS$RMT1:
$ DEALLOCATE _MYSYS$RMT1:
```

In this example, the host BOS.EXAMPLE.COM contains the tape drive. The two colons separating the host name follow the style of DECnet device specifications; RMTALLOC accepts either single or double colon separators. MYTAPE is the tape logical name associated with the pseudo device. In the MOUNT statement, the /FOREIGN qualifier specifies that the device is not file structured.

## RMTALLOC CD-ROM Access Examples

In the first example, a CD-ROM on an OpenVMS host is accessed from another OpenVMS host:

```
$ RMTALLOC/CD/NOWRITE CONTROL::DISK$CD: MYCD /USERNAME=SYSTEM
%RMT-I-ALLOC, _MYSYS$RCD3: allocated (BOS.EXAMPLE.COM::DISK$CD:)
$ MOUNT/OVERRIDE=ID MYCD:
%MOUNT-I-WRITELOCK, volume is write locked
%MOUNT-I-MOUNTED, VMS055LST1 mounted on _MYSYS$RCD3:
```

Read from the CD-ROM drive, then dismount and deallocate the drive:

```
$ DISMOUNT MYCD:
$ DEALLOCATE MYCD:
```

In this example:

- The RMTALLOC statement includes the /CD qualifier to indicate the device is a CD-ROM drive.
- The /NOWRITE qualifier is the default whenever you specify /CD; omitting it indicates the device is read-only.
- CONTROL: is the host on which the CD-ROM is located.
- DISK\$CD: is the drive name.
- MYCD is the device name.
- The /USER=SYSTEM qualifier ensures the SYSTEM account is accessed on the remote host. (If the remote host is an OpenVMS system, the account used must have LOG\_IO privilege.)
- The /MOUNT command uses the /OVERRIDE=ID qualifier to inhibit MOUNT protection checks of the volume identifier in the CD-ROM label.
- The DISMOUNT and DEALLOCATE commands are used after information is read from the CD-ROM.

In the next example, a CD-ROM drive on a UNIX host is accessed:

```
$ RMTALLOC/CD/NOWRITE UNIXBOX::"/dev/rsr1" MYCD /USERNAME=root
%RMT-I-ALLOC, _MYSYS$RCD3: allocated (UNIXBOX.EXAMPLE.COM::/dev/rsr0)
```

```
$ MOUNT/OVERRIDE=ID MYCD:
%MOUNT-I-WRITELOCK, volume is write locked
%MOUNT-I-MOUNTED, VMS055LST2 mounted on _MYSYS$RCD3:
```

Read from the CD-ROM drive, then dismount and deallocate the drive:

```
$ DISMOUNT MYCD:
$ DEALLOCATE MYCD:
```

In this example, the RMTALLOC statement contains the name of the UNIX host (UNIXBOX) that has the CD-ROM drive. The device name is specified in UNIX style. If the device name is not specified, the default is the `/dev/rsr0` device.

The user name is set to the `root` login, which is a UNIX login similar to the OpenVMS SYSTEM login. The MOUNT, DISMOUNT, and DEALLOCATE commands are the same whether the CD-ROM is on another OpenVMS system or a UNIX host.

## Using RMTALLOC Qualifiers

RMTALLOC qualifiers, apart from those already discussed (`/CD`, `/NOWRITE`, and `/USERNAME`), provide the following additional features:

- VMS-to-VMS negotiation
- Remote operator interaction
- Remote login control
- Message suppression
- Write protection

## VMS-to-VMS Negotiation

When accessing a drive on an OpenVMS host, if both systems are running MultiNet, RMT uses an improved protocol to transfer OpenVMS device attributes and I/O completion status values between your system and the remote host. Because this negotiation is compatible with UNIX Operating System implementations of RMT (including BSD and SunOS), it is enabled by default. You may disable it with the RMTALLOC `/NOVMS_ATTRIBUTES` qualifier if compatibility problems arise.

## Suppressing Messages

Use the `/NOLOG` qualifier to suppress system status messages. Use this option in DCL command procedures to prevent the messages from displaying. `/LOG` is the default.

# Controlling Remote Login

Use the `/PASSWORD` qualifier to specify a password for the remote host when you do not have a `.RHOSTS` file. This qualifier poses a security risk because the password is transmitted over the network as plain text. When you use `/PASSWORD`, the REXEC server (instead of the RSH shell server) is called on the remote host. The password is in the format used by the system you are contacting.

Similarly, use the `/USERNAME` qualifier to specify the login name to access on the remote system. On a UNIX system, the specified login must exist in the `/etc/passwd` file.

Use the `/TRUNCATE_USERNAME` qualifier to truncate an OpenVMS user name to a maximum of eight characters for use with some UNIX systems.

# Interacting with the Remote Operator

Use the `/SEMANTICS` qualifier only with tape drive access to interact with the operator of the remote system or to specify tape drive information to the remote system.

Use the optional `BLOCKSIZE` and `DENSITY` values to specify information used by the remote system to read the tape. All other values send messages to the operator via the OPCOM message facility. Without specifying any values, the following information is displayed when `RMTALLOC` is called:

```
%%%%%%%%%%%%% OPCOM  date  time  %%%%%%%%%%%%%%  
FROM NODE nodename AT date time  
REQUEST nn, FROM USER username ON nodename  
Please mount device _nodename$devicename:  
RMT tape service request from nodename.domain
```

The `COMMENT` value is specified as a string enclosed in double-quotes; the information is displayed in the remote OPCOM message, either appended to or replacing the default text depending on whether the resulting length is less than the 78-character maximum. Supplying the `COMMENT` value is the only way you can send a tape-specific message to the remote operator. The OPCOM message from the `DCL MOUNT/COMMENT` command is not passed to the remote RMT server; this message is only sent to OPCOM for a local operation. The default `RMTALLOC` command mounts the remote tape foreign, causing an OPCOM message to be generated if the tape drive is offline.

**Note:** The `RMTALLOC /SEMANTICS=NOMOUNT` command does not work correctly with multivolume BACKUP savesets.

# Write Protection

Use the `/WRITE` qualifier to make sure that write protection is respected; `/NOWRITE` is the default for CD-ROM drives.

By default, `RMTALLOC` mounts a remote tape drive for read-write access. If the remote tape drive is physically protected from write access, you must use `/NOWRITE` to indicate you want read-only access to the tape drive. Otherwise, the remote UNIX RMT server usually returns an error indicating "Permission Denied."

# 15. Configuring and Managing FTP

The FTP (File Transfer Protocol) server provides file access between remote systems. Information about using this utility is in the *MultiNet User's Guide*.

FTP is configured automatically during the MultiNet installation procedure. This chapter explains how to administer the FTP client and server.

## Configuring the FTP Client

Configuring the FTP client consists of creating a `MULTINET:FTP.INIT` file for site-specific purposes. When the FTP client is started, the commands in the `MULTINET:FTP.INIT` file are executed before the commands in the `SYS$LOGIN:FTP.INIT` file of the user running FTP. See the *MultiNet User's Guide* for more information about creating `FTP.INIT` files.

The FTP client censors the output of the `NLST/LIST` commands. A period (.) replaces unprintable characters.

If the logical name `MULTINET_FTP_DELAY_TRANSFER_NEGOTIATION` is defined, then the FTP client does not attempt to negotiate `STRU O VMS` transfer mode until after you have logged into the remote system successfully. You can define this logical at the user or system level.

```
$ DEFINE MULTINET_FTP_DELAY_TRANSFER_NEGOTIATION anything
```

If the logical `MULTINET_FTP_SIZE_BEFORE_GET` is defined to `FALSE`, `NO`, or `0` (zero) the `SIZE` command will not be sent before the `GET` command for a file. When the logical is not defined, or is defined to a value other than `FALSE`, `NO`, or `0`, the `SIZE` command is sent. Any returned value is used to preallocate the file size and to report progress of a file transfer. Some FTP servers leave the file open accidentally after the `SIZE` command.

If the logical name `MULTINET_FTP_NONPASV` is defined, then the FTP client will start up in `PASSIVE OFF` mode. The default client behavior is `PASSIVE ON`.

# Managing an FTP Server

Managing an FTP server may include the following tasks:

- Creating an anonymous FTP login (see the *Configuring Anonymous FTP* section).
- Creating an FTP server login command procedure (see the *Creating an FTP Server Login Command Procedure* section).
- Using log files (see the *Using FTP Log Files* section).
- Managing FTP security (see the *Managing FTP Security* section).
- Specifying a message at connect time (see the *Specifying a Message at Connect Time* section).
- Specifying UNIX-style listings (see the *Specifying UNIX-Style Listings* section).
- Specifying the maximum idle time before a connection times out (see the *Specifying the Maximum Idle Time* section).
- Using FTP server site commands (see the *Using FTP Site Commands* section).
- Using Network Service Monitoring (see the *Network Service Monitoring* section).
- Using Session Accounting (see the *Session Accounting* section).

## Configuring Anonymous FTP

To set up anonymous FTP access on your system:

1. Create an account named ANONYMOUS with the password GUEST. Any password works from the remote host but the account is validated with the password GUEST. Use the OpenVMS AUTHORIZE utility to create the account:

```
$ RUN SYS$SYSTEM:AUTHORIZE
UAF>ADD ANONYMOUS /PASSWORD=GUEST /OWNER="Anonymous FTP" -
/DEVICE=device/UIC=uic
UAF> Ctrl+Z
$
```

*uic* is the UIC to use for ANONYMOUS.

*device* is the device on which the directory [ANONYMOUS] is located.

2. Use the NOLOCAL, NOBATCH, NOREMOTE, and NODIALUP access restrictions to the ANONYMOUS login to prevent other forms of access. You set these restrictions by running AUTHORIZE and issuing the command:

```
UAF>MODIFY ANONYMOUS /NOLOCAL /NOBATCH /NOREMOTE /NODIALUP
```



3. To prevent access to the account through DECnet, do not grant the NETMBX privilege to ANONYMOUS. To make sure that ANONYMOUS does not have the NETMBX privilege, issue the following AUTHORIZE command:

```
UAF>MODIFY ANONYMOUS /PRIV=NONETMBX /DEFPRIV=NONETMBX
```

4. To restrict anonymous FTP access to the [ANONYMOUS] directory tree, use the NET-CONFIG utility SET ANONYMOUS-FTP-DIRECTORY and SET ANONYMOUS-FTP-ACCESS to set this access restriction. See the *MultiNet Administrator's Reference* for additional information about NET-CONFIG commands.

Anonymous FTP server processes are created with the process name \*FTP\_ *pwd*,  
*pwd* is the password the user specifies.

By convention, many people specify GUEST, their personal name, or their local user name for the password, because anything is accepted.

If you do not want to create FTP\_SERVER.LOG files in the anonymous directory, assign a new default directory for the login with a directory restriction to make sure the log files appear in the correct directory. In this example, an alternate FTP directory is created for the log files:

```
$ SET DEFAULT SYS$SYSTEM
$ RUN SYS$SYSTEM:AUTHORIZE
UAF>MODIFY ANONYMOUS/DEVICE=SYS$COMMON:/DIR=[SYSMGR.ANONYMOUS]
UAF>EXIT
$ CREATE/DIRECTORY/OWNER=ANONYMOUS SYS$COMMON:[SYSMGR.ANONYMOUS]
$ MULTINET CONFIGURE
NET-CONFIG>SET ANONYMOUS-FTP-DIRECTORY USERS:[ANONYMOUS]
NET-CONFIG>EXIT
$
```

You can now control the setting of the MULTINET\_ANONYMOUS\_FTP\_CONTROL logical name using either the network configuration utility (MULTINET CONFIGURE):

```
$ MULTINET CONFIGURE
NET-CONFIG>SET ANONYMOUS-FTP-ACCESS NOWRITE,NOSPAWN
NET-CONFIG>EXIT
[Changes take effect after the next MultiNet reload]
$
```

This is a new method for controlling the definition of that logical name. Other methods required you to define the logical name yourself during system startup. You can make the change take effect before the next system reboot by defining the associated logical name DEFINE /SYSTEM /EXECUTIVE\_MODE MULTINET\_ANONYMOUS\_FTP\_CONTROL " . . . ". For example:

```
$ DEFINE /SYSTEM /EXECUTIVE MODE MULTINET_ANONYMOUS_FTP_CONTROL -
_$ "NOSPAWN,NODELETE"
```

The default setting is NOWRITE, NOSPAWN. See the table below for other options.

If you do not want to use the “anonymous” name, there is a logical that will allow users to use names which are not anonymous, but have the same anonymous account behavior. The MULTINET\_ANONYMOUS\_USERNAMES logical usage is shown in the following example:

```
$ DEFINE /SYSTEM /EXEC MULTINET ANONYMOUS USERNAMES
_$ "anonymous,user1,user2,..."
```

If you define this logical as shown in the preceding example and set the “user1,user2,...” accounts using the same password as the anonymous account, then the FTP server will treat “user1,user2,...” as an anonymous type of user.

## Specifying a Range of FTP Server Port Numbers

The logical MULTINET\_FTP\_SERVER\_DATA\_PORT\_RANGE specifies the range of port numbers to use for passive connections. The format is:

```
$ DEFINE /SYSTEM /EXEC MULTINET FTP_SERVER_DATA_PORT_RANGE "<starting port number><end port number>"
```

When this logical is defined, the FTP server will use port numbers between the specified values for the data channel when operating in passive mode.

## Creating an FTP Server Login Command Procedure

To limit user activities during an FTP session, edit the FTP\_SERVER.COM file using this command:

```
$ MULTINET FTP/SERVER [qualifier]
```

The table below lists the FTP server qualifiers.

Qualifier	Purpose
/ACCESS=( [NOLIST], [NOWRITE], [NOSPAWN], [NOREAD], [NODELETE] )	Denies write or search access for this session; NOSPAWN disables the site SPAWN command; NOREAD prohibits read access. By default, /ACCESS=NOSPAWN is used for anonymous FTP sessions to prevent users from spawning commands. NOWRITE disables the storing of files. NODELETE disables the deletion/renaming of files.
/DIRECTORY=(directory1,...)	Restricts access to these directory trees (note plural).

/GET_REMOTE_INFO	<p>Gets information about the remote system. This qualifier works by defining the logical names MULTINET_FTP_ADDRESS, MULTINET_FTP_HOSTNAME, MULTINET_FTP_LOCAL_ADDRESS, and MULTINET_FTP_ANONYMOUS_PASSWORD and then exiting without invoking the FTP server.</p> <div data-bbox="669 600 1469 856" style="background-color: #e6f2ff; padding: 10px; border-left: 3px solid #0070c0;"> <p><b>Note:</b> MULTINET_FTP_ANONYMOUS_PASSWORD is only set if the user name is "anonymous."</p> </div> <p>When MULTINET_FTP_DONT_REPORT_FILESIZE is defined, the estimate of the number of bytes to be transmitted is not included in the 150 reply line to a GET operation.</p>
/MESSAGE= <i>message</i>	Displays a banner message when the user logs in. This message precedes the "User xxx logged in..." line.
/REJECT	Instead of accepting the connection, rejects the login with the error specified in the /MESSAGE qualifier.

## Using FTP Log Files

The MultiNet FTP server keeps a log of all FTP transactions that occur after login between the client and server in an FTP\_SERVER.LOG file in the user's login directory on the server system. A log file is created for each FTP client session. The previous log is overwritten when a new session starts, but you can specify a number of log files to retain.

**Note:** If the MultiNet FTP server process does not start or mysteriously disappears, examine the beginning of FTP\_SERVER.LOG for error messages. Because the system-wide login command

procedure (SYS\$MANAGER:SYLOGIN.COM) and the user's LOGIN.COM are executed as part of the server process creation, errors in these procedures can cause the server process to terminate. In most instances, however, the reason for the process terminating appears at the beginning of the FTP\_SERVER.LOG file.

The following sample log file contains the FTP transactions involved when the user logs in under the user name HOLMES, issues a DIRECTORY command, and then retrieves the file FOO.BAR.

```
-----  
FTP Login request received at Wed Jun 14 19:05:10 2015  
from remote IP address 127.0.0.1  
-----  
>>> 230 User HOLMES logged into U1:[HOLMES] at Wed 07-Jun-2019 19:05, job  
3a.  
<<< TYPE A  
>>> 200 Type A ok.  
<<< STRU F  
>>> 200 Stru F ok.  
<<< MODE S  
>>> 200 Mode S ok.  
<<< PORT 127,0,0,1,4,14  
>>> 200 Port 4.14 at Host 127.0.0.1 accepted.  
<<< LIST  
>>> 150 List started.  
>>> 226 Transfer completed.  
<<< PORT 127,0,0,1,4,15  
>>> 200 Port 4.15 at Host 127.0.0.1 accepted.  
<<< RETR foo.bar  
>>> 150 ASCII retrieve of USERS:[HOLMES]FOO.BAR;1 started.  
>>> 226 Transfer completed. 210 (8) bytes transferred.  
<<< QUIT  
>>> 221 QUIT command received. Goodbye.  
HOLMES job terminated at 11-JUN-2019 19:05:23.08
```

By setting the logical name MULTINET\_FTP\_SERVER\_LOG\_LIMIT in the LOGIN.COM file, you can specify that log files be retained. Set the logical name to a dash (-) to retain all log files, or specify a number in the range of 1 to 32000.

Directory size restrictions limit the number of potential files that can actually be created. If you do not specify a number or value, one log file is created or overwritten for each FTP session. Use the DCL PURGE command to delete unneeded log files. The following example specifies that 42 log files be retained:

```
$ DEFINE MULTINET_FTP_SERVER_LOG_LIMIT 42
```

# Managing FTP Security

Because the FTP server process is started by running `SYS$SYSTEM:LOGINOUT.EXE`, both the system-wide login command procedure (`SYS$MANAGER:SYLOGIN.COM`) and the specific user's `LOGIN.COM` are executed. As a result, any customization (default file protection, process/job logical name definitions, and so on) done in these command procedures is available under the FTP server process.

All standard OpenVMS security-checking mechanisms also validate the FTP server process creation. If a login command procedure contains any commands specific to interactive jobs (for example, `SET TERMINAL` commands), the FTP server process may crash. To avoid this problem without altering the functionality of command procedures, use the DCL lexical function `F$MODE` with interactive commands. For example:

```
$ IF F$MODE() .EQS. "INTERACTIVE" THEN SET TERMINAL /INQUIRE
```

You can use the following logicals in the `FTP_SERVER.COM` command procedure to restrict specific users from some types of access:

- The following logical restricts `username` to accessing only the specified directories when connecting to the host using FTP:

```
MULTINET_username_FTP_DIRECTORY "directory-spec,..."
```

This logical is used in the `FTP_SERVER/DIRECTORY=directory-spec,...` qualifier.

- The following logical restricts `username` to only the type of access specified when accessing the host via FTP:

```
MULTINET 'username'_FTP_CONTROL "access-spec,..."
```

This logical is used in the `FTP_SERVER/ACCESS=access-spec,...` qualifier.

```
access-spec=[NO]LIST, [NO]WRITE, [NO]SPAWN, or [NO]READ
```

- The following logical limits the information given out on connection or when using the `STAT` command:

```
MULTINET_FTP_CONNECT_BANNER "FTP server name"
```

If this logical is defined as whitespace, operating system and TCP stack information is removed from the FTP server connection banner. If this logical is defined with a specific FTP server name, the information banner does not appear in response to the `STAT` command.

## Accepting Wildcards upon Delete

You can apply the logical `MULTINET_FTP_DISALLOW_WILDCARD_DELETES` to anything to disallow the new functionality of accepting wildcards on delete. This may be done at the process, group, or system level.

## Specifying a Message at Connect Time

The `MULTINET_FTP_ANNOUNCE` logical provides a `SYS$ANNOUNCE`-style message along with the "220" banner at connect time. Define the logical in a fashion similar to `SYS$ANNOUNCE`, using one of the following commands:

```
$ DEFINE/SYSTEM MULTINET_FTP_ANNOUNCE "message text"
```

In the following version, the announcement is in the specified file:

```
$ DEFINE/SYSTEM MULTINET_FTP_ANNOUNCE "@file specification"
```

## Specifying UNIX-Style Listings

If you define the logical name `MULTINET_FTP_UNIX_STYLE_BY_DEFAULT`, the FTP Server starts in UNIX emulation mode.

The control of version number displays has been reworked in response to `LIST` and `NLST` commands. The default is VMS-mode output.

The logical name `MULTINET_FTP_UNIX_STRIP_VERSION` no longer has any effect. In UNIX mode, the FTP Server always removes version numbers from directory listings.

The logical name `MULTINET_FTP_STRIP_VERSION` causes VMS mode output to have no versions.

**Note!** It is recommended that you NOT use the `MULTINET_FTP_STRIP_VERSION` logical. Stripping version numbers from the VMS mode `LIST` output can cause problems for some FTP clients (notably `WS_FTP`).

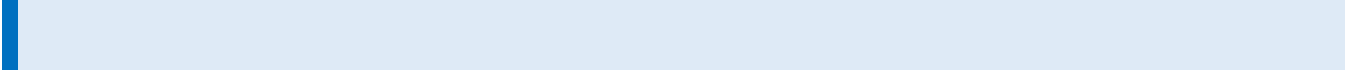
The logical name `MULTINET_FTP_ALL_VERSIONS` requests the `NLST` and `LIST` commands to display all version numbers. If `MULTINET_FTP_ALL_VERSIONS` is defined, the logical name `MULTINET_FTP_STRIP_VERSION` has no effect.

**Note:** MULTINET\_FTP\_ALL\_VERSIONS is ignored if the FTP Server is in UNIX emulation mode.

The FTP Server updates UNIX emulation improving MultiNet interoperability with various FTP clients. Features of the UNIX emulation mode are

- The syntax you use for a directory determines the mode you want. For example, `CWD /` uses UNIX mode; `CWD []` uses VMS mode.
- The `LIST` command returns output similar to that produced by `ls -al`.
- The logical name `MULTINET_FTP_UNIX_STYLE_CASE_INSENSITIVE` allows UNIX style filename handling to be case insensitive.
- Mixed case file names and those with special characters are translated into legal OpenVMS file names using the NFS mappings.
- The directories listing uses UNIX syntax. For example, `USERS: [MRUHL]` becomes `/users/mruhl`.
- When changing directories or referencing files using an absolute UNIX pathname, directory lookups treat `SYS$LOGIN` as if they were the root directory (`/`). So, if `SYS$LOGIN` is `USERS: [MRUHL]`,  
  
`/login.com` refers to `USERS: [MRUHL] LOGIN.COM` and  
`/multinet_common_root/multinet` refers to  
`USERS: [MRUHL.MULTINET_COMMON_ROOT.MULTINET]` if that directory exists. If it does not exist, the first segment of the pathname is used as the device specification in a second lookup attempt, and `/multinet_common_root/multinet` refers to `MULTINET_COMMON_ROOT: [MULTINET]`.
- If the FTP server is in UNIX mode, the `SYST` command displays the banner “UNIX MultiNet Unix Emulation.” If the FTP server is in VMS mode, the `SYST` command displays the equivalence string associated with the `MULTINET_FTP_SYST_BANNER` logical name (if defined). Otherwise, the `SYST` command displays “VMS MultiNet Vx.y(rev)”:
  - `Vx.y` is the MultiNet version number.
  - `(rev)` is the revision number of the FTP server.

**Note:** The logical name `MULTINET_FTP_SYST_BANNER` is ignored if the FTP Server is already in UNIX mode.



The FTP server does not drop spaces in filenames. It converts them to the character sequence \$7A.

The FTP server protects privileged ports by not opening data connections to privileged ports.

The file open routines allow all modes to fetch data from a file open for write but marked for shared access.

The FTP service corrects synchronization problems resulting in messages repeatedly sent to the FTP client.

There is no PASV command interference with data link window sizing.

If you want the device name, the file name, and the directory name included in the results of all NLST commands, define the logical MULTINET\_FTP\_INCLUDE\_DEVICE\_IN\_NLST. This logical may be declared system wide or in the user's LOGIN.COM file.

The FTP Service corrects a problem with RENAME operations with UNIX-style file specifications. The RENAME operation overrides the current protection of the file to do the operation and then restores it afterwards. This is necessary because directories are created such that they cannot be deleted without changing the protection. To cause RENAME to observe the file protection, define the logical MULTINET\_FTP\_OBSERVE\_VMS\_PROTECTION to true.

## UNIX File Names

An FTP default is to rename files by changing the final dot (.) to \$5N. The logical MULTINET\_FTP\_DODROP1DOT lets you override this FTP default by dropping the final dot and NOT adding \$5N.

VMS always implies that a dot is present in file names regardless of whether it is followed by an extension. VMS also does not support multiple dots in a file name. The rule FTP follows is that when there is only one dot, and that dot is the final character, the dot is converted to \$5N. The resultant local file is then distinguishable from a similarly named file that did not have a dot. For example, "FILE." becomes "FILE\$5N" when using the FTP default; however, "FILE." becomes "FILE" with the logical defined.



The FTP server displays the creation month, day, and year of a file for a UNIX mode directory if the file is older than 1 year (365 days). If the logical `MULTINET_FTP_UNIX_YEAR_OLD_FILES` is defined False, No, or 0 (zero), the old behavior is restored, displaying all files with Month, Day, and Time.

The logical `MULTINET_FTP_DISALLOW_UNIX_STYLE` controls whether UNIX-style filename parsing is done. The default value for `MULTINET_FTP_DISALLOW_UNIX_STYLE` is true (T), UNIX-style filename parsing is not handled. If you want UNIX-style filename parsing, you must define this logical as FALSE. When UNIX-style parsing is enabled, it is not normally done until a CD command has been done with a directory specification that contains a “/” in it. For example:

```
FTP> cd ../my_directory
```

**Note:** For some FTP clients (MultiNet is one of them) you will have to enclose the directory specification in quotes (“ ”) when it contains the “/” to prevent the client from attempting to parse it.

To exit UNIX-type filename parsing, use a `cd` command with either the “[” or “<” character in the directory specification. For example:

```
FTP> cd [-.my_directory]
```

```
$ DEFINE/SYSTEM/NOLOG/EXEC MULTINET_FTP_DISALLOW_UNIX_STYLE FALSE
```

Some graphical display FTP clients expect the output of directory commands to be in a UNIX system format. To enable this UNIX format, use the following either at the system level or in the user's LOGIN.COM:

```
$ DEFINE MULTINET_FTP_DISALLOW_UNIX_STYLE FALSE
```

and

```
$ DEFINE MULTINET_FTP_UNIX_STYLE_BY_DEFAULT ANYTHING
```

## Specifying the Maximum Idle Time

The `MULTINET_FTP_MAXIMUM_IDLE_TIME` logical specifies the length of time before an idle FTP server connection times out. The value is set in seconds, with a default of 300 seconds. If this logical is set to 0, timeouts are disabled.

The logical name `MULTINET_FTP_FAST_TIMEOUT` is equivalent to the settings in the logicals `MULTINET_NAMESERVER_RETRANS` and `MULTINET_NAMESERVER_RETRY` for the FTP server

process to 5 and 2 respectively. This helps the FTP server start up faster when DNS PTR records for the client are improperly defined or nonexistent.

## Using FTP Site Commands

The table below lists the commands for controlling and configuring the FTP server from the FTP prompt.

Command	Description
<code>SITE SHOW TIME</code>	Shows the time on the system the FTP server is running on.
<code>SITE NONE</code>	This is a NO OPERATION.
<code>SITE PRIV [privileges]</code>	Turns ON or OFF VMS privileges. If no privileges are specified, displays the current privileges.
<code>SITE RMS BLOCK [ON OFF]</code>	Turns Block Image mode transfers ON or OFF, or if there is no argument, displays the current state.  When image (binary) transfers are done with Block Image mode OFF (the default), the FTP server opens VMS record files such that the record control information is not included in the data and a linefeed separates each record. With Block Image mode ON the record control information and the data are transferred.
<code>SITE RMS RECSIZE [value]</code>	With no argument, displays the current record size. With an argument, sets the record size for Image PUT transfers to the size specified. The default is 512.
<code>SITE SPAWN command</code>	Spawns a subprocess and uses the rest of the line as a VMS DCL command. Not valid for CAPTIVE processes.
<code>SITE RMS STREAM [ON OFF]</code>	With no argument, displays the current state. When ON, writes text files in Stream-LF format.

	When OFF (default), writes text files as VMS carriage-control record files.
SITE +VMS+	Enables VMS + mode transfers.
SITE VMS	Disables VMS + mode transfers.
SITE WINDOW-SIZE	With no argument, displays the TCP window size. With an argument, sets the window size for the data channel. Default value is 32768 bytes

## Defining FTP Messages

The `MULTINET_FTP_221_REPLY` logical lets you define the message users see when they end the FTP session. If you do not define this logical, MultiNet uses the default message instead “221 QUIT command received. Goodbye.” You can define a text string or file. If the string starts with the “at” sign @, it specifies the name of a file containing text to be displayed. For example:

```
$ DEFINE/SYSTEM/EXEC MULTINET_FTP_221_REPLY -
_$ "Connection to FTP server has been closed"
```

Now, when the user closes the FTP connection, the following message appears:

```
221 Connection to FTP server has been closed
```

The `MULTINET_FTP_230_REPLY` logical defines a message to appear when a user successfully logs in. If you do not define this logical, MultiNet uses the default message instead. As with `MULTINET_FTP_221_REPLY`, you can define a text string or file. For example:

```
$ DEFINE/SYSTEM/EXEC MULTINET_FTP_230_REPLY "Login successful"
```

Now, when the user logs in using FTP, the following message appears:

```
230 Login successful
```

The `MULTINET_FTP_ANONYMOUS_230_REPLY` logical defines a message to appear when an ANONYMOUS user successfully logs in. If you do not define this logical, MultiNet uses the default message instead. As with `MULTINET_FTP_230_REPLY`, you can define a text string or file. For example:

```
$ DEFINE/SYSTEM/EXEC MULTINET_FTP_ANONYMOUS_230_REPLY -
_$ "ANONYMOUS login successful"
```

Now, when a user logs in using the ANONYMOUS account, the following message appears:

```
230 ANONYMOUS login successful
```

The MULTINET\_FTP\_421\_REPLY logical lets you define the message users see when they connect to the server but should not log in. After sending the message, the connection closes. For example, you can define this logical to prevent FTP access for a short time period. Be sure to deassign the logical after this period to allow FTP access again. If the string starts with the “at” sign @, it specifies the name of a file containing text to be displayed. For example:

```
$ DEFINE/SYSTEM/EXEC MULTINET_FTP_421_REPLY -  
_ $ "System maintenance in progress until 17:30"
```

Now, when the user connects to the host using FTP, the following message appears and then the connection closes:

```
421 System maintenance in progress until 17:30
```

## Specifying the Name of a Log File

The MULTINET\_FTP\_LOGFILE (system level, executive mode) logical can be defined to specify the name of a log file. For example:

```
$ DEFINE/SYSTEM/EXEC MULTINET_FTP_LOGFILE -  
_ $ SYS$COMMON:[SYSMGR]FTPLOGIN.LOG
```

If this logical exists, the FTP server writes a record to the specified file each time a user attempts to log in. Each record includes the date and time, the remote host's internet address, and whether the login succeeded. This is especially useful to use if you suspect someone has tried to break into the FTP server.

This logical specifies the name of the file to which *all* commands and responses to ANONYMOUS FTP services are logged. If MULTINET\_FTP\_LOG\_ALL\_USERS is also defined, then commands and responses for all users are logged. If MultiNet is already running, the Master Server must be restarted (@MULTINET:START\_SERVER) for these definitions to take effect.

The logical MULTINET\_FTP\_LOG\_ALL\_USERS causes all commands and responses to be logged to the file defined by MULTINET\_FTP\_LOGFILE. The default (when this logical is not defined) is to just log the commands and responses for anonymous users.

```
$ DEFINE/SYSTEM/EXEC MULTINET_FTP_LOG_ALL_USERS TRUE
```

The FTP client and the FTP server normally check the record size of an ASCII transfer and disallow more than 8192 byte records (as a sanity check). However, you can define the MULTINET\_FTP\_MAXREC logical to override the default of 8192. The definition of the MULTINET\_FTP\_MAXREC logical is commented out but defined in the FTP\_CONTROL.COM file as follows:

```
$ DEFINE/SYSTEM/NOLOG/EXEC MULTINET_FTP_MAXREC 8192
```

Note that the maximum record size supported by OpenVMS is 65535.

## Defining a File Name

The `MULTINET_DIRECTORY_MESSAGE_FILENAME` logical can be defined to name the file you want to appear when a session enters a directory. For example:

```
$ DEFINE/SYSTEM/EXEC MULTINET_DIRECTORY_MESSAGE_FILENAME example.txt
```

## Password Lifetime Warnings

This section describes how to define password messages in MultiNet.

### Defining Password Messages

The `MULTINET_FTP_PASSWORD_WARNING_MESSAGE` logical lets you define the message users see when their password is going to expire. If you want the amount of time before the password expires to be included, add `%s` to the logical.

```
$ DEFINE/SYSTEM/EXEC MULTINET_FTP_PASSWORD_WARNING_MESSAGE %s  
$ DEFINE/SYSTEM/EXEC MULTINET_FTP_PASSWORD_WARNING_MESSAGE message text  
string
```

The `MULTINET_FTP_PASSWORD_WARNING_TIME` logical uses the VMS delta time to specify the minimum remaining lifetime for the user's password. If the remaining lifetime is greater than the VMS delta time then no message appears. It is necessary to define this value to enable checking for the remaining lifetime of a password.

```
$ DEFINE/SYSTEM/EXEC MULTINET_FTP_PASSWORD_WARNING_TIME dddd hh:mm:ss.hh
```

The `MULTINET_FTP_RECEIVE_THRESHOLD` logical specifies the amount of buffer space that can be used to buffer transmitted data on the data socket. The default value is 6144. If this logical is defined and it begins with a `/`, then it specifies the fraction of the window size; if only a fraction is specified, then it indicates the number of bytes to be used. The `?` in the logical represents where defined values go.

```
$ DEFINE/SYSTEM/EXECUTIVE MULTINET_FTP_RECEIVE_THRESHOLD ?
```

The `MULTINET_FTP_NOLOGIN_EXPIRED_PASSWORD` logical lets you prevent accounts with expired passwords from logging in.

```
$ DEFINE/SYSTEM/EXEC MULTINET_FTP_NOLOGIN_EXPIRED_PASSWORD TRUE
```

will prevent a user with an expired password from logging in and displays the following message:

<Your password has expired; contact your system manager>

## Checking IP Address

By default the MultiNet FTP server checks the IP address given in the port command and does not make the connection if the IP address does not match that of the control connection. This can be disabled by defining the logical `MULTINET_FTP_SERVER_RELAXED_PORT_COMMAND`.

```
$ DEFINE MULTINET_FTP_SERVER_RELAXED_PORT_COMMAND 197.0.0.1
```

# Configuring the FTP server for TLS (FTPS)

Follow these steps to configure the MultiNet FTP server to allow TLS authentication:

- Generate or obtain certificate and key files. On OpenVMS V8.3 and higher `SSL$COM:SSL$CERT_TOOL` can be used to do this.
- Place the certificate and key file where you want them and verify that the protection is set such that world has no access.
- Using `MULTINET CONFIGURE/SERVER` select FTP and set the `RFC-4217-CERTIFICATE` and `RFC-4217-KEY` parameters to the location of the appropriate files. Optionally set the `REQUIRE_TLS` parameter to YES.
- Restart the MultiNet master server - `@multinet:start_server restart`

## FTP server parameters for TLS

`RFC4217-CERTIFICATE` Specifies the certificate file to be used with RFC 4217 negotiation. The certificate and key files must be created by an external means such as the SSL certificate tool and be in PEM format. Both a certificate and key file must be specified set up to allow TLS negotiation. On OpenVMS V8.3 and higher you can use `@SSL$COM:SSL$CERT_TOOL`.

`RFC4217-KEY` Specifies the private key file to be used with RFC 4217 negotiation. The certificate and key files must be created by an external means such as the SSL certificate tool and be in PEM format. Both a certificate and key file must be specified set up to allow TLS negotiation. On OpenVMS V8.3 and higher you can use `@SSL$COM:SSL$CERT_TOOL`.

REQUIRE-TLS YES Specifies that user authentications other than anonymous and users that have no password must use TLS authentication. The FTP USER command will get a 530 response if it is issued before TLS authentication has been done. This prevents passwords from being exchanged in clear text mode with the users.

## Network Service Monitoring

FTP's network service monitoring is based on RFC 2788 (Network Services Monitoring MIB). Information is maintained only while the service is active. The following items from the Network Services Monitoring MIB (RFC 2788) are available in the enterprises.105.2.25 MIB:

ApplAccumulatedInboundAssociations	(Counter) the total number of connections that the FTP Listener program has serviced since it was started. enterprises.105.2.21.10
ApplDescription	(String) Description of the program/application. This is the banner that gets printed when the client connects to the FTP Listener program. enterprises.105.2.21.16
ApplInboundAssociations	(Counter) The number of connections currently active. enterprises.105.2.21.8
ApplIndex	(Integer) unique application index. The port FTP is offered on (21). enterprises.105.2.21.1
ApplLastChange	(TimeTicks) the value of sysUpTime when the FTP Listener program entered the current state. enterprises.105.2.21.7
ApplLastInboundActivity	(TimeTicks) the value of sysUpTime at the time the most recent connection was established. enterprises.105.2.21.12
ApplName	(String) FTP. enterprises.105.2.21.2

ApplOperStatus	(Integer) the operational status of the FTP Listener program; the values are: up(1), down(2), halted(3), congested(4), restarting(5), quiescing(6). Some of these values may not be used. enterprises.105.2.21.6
ApplRejectedInboundAssociations	(Counter) the number of connections that have been rejected (due to not being allowed from the access list values). enterprises.105.2.21.14
ApplUptime	(TimeTicks) the value of the SNMP variable sysUpTime when the FTP Listener program was started. This time has a resolution of 5 minutes. enterprises.105.2.21.5
ApplVersion	(String) the version of the FTP Listener program. enterprises.105.2.21.4

This feature requires the SNMP Agent X functionality. To use this SNMP must be configured to have Agent X service enabled, and to allow the system's IP and the local host addresses (127.0.0.1) to each be an AGENTX\_PEER. See Chapter 23 for more information on SNMP and Agent X. This information can be displayed with the MULTINET SHOW/SNMP command and can be displayed with a MIB browser.

To enable network service monitoring, do the following:

```
$ MULTINET CONFIGURE /SERVER
SELECT FTP
SET FLAGS SNMP_MONITORED | PASS FOREIGN SOCKET
WRITE
EXIT
$ @MULTINET:START_SERVER
```

Any service using TCP\_INIT, TCP\_LISTEN, and TCP\_CONNECTED routines may use SET FLAGS SNMP\_MONITORED. The level of functionality may vary with the service.

## Session Accounting

MultiNet can record accounting information from services that have been enabled. Currently this includes FTP and SMTP. The accounting information includes information about when a network session took place and how much data was transferred. The accounting facility is enabled by setting the accounting port and the accounting host and reading MULTINET:ACCOUNTING.CONF for additional



configuration information. The format of the accounting records is described in `MULTINET_ROOT:[MULTINET.EXAMPLES]ACCOUNTING.H`

A sample program using this is in `MULTINET_ROOT:[MULTINET.EXAMPLES]ACC_DUMP.C`

You must configure FTP and session accounting in order to activate the accounting function. `FTP-ACCOUNTING-HOST` is the name of the system running the accounting program. `FTP-ACCOUNTING-PORT` is the port number that the program was set up to listen on. FTP accounting can be configured with the following:

```
$ MULTINET CONFIGURE /NETWORK
  SET FTP-ACCOUNTING-HOST name
  SET FTP-ACCOUNTING-PORT number
  WRITE
  EXIT
```

In order for accounting to be activated before your next reboot, you can define the logicals as follows:

```
$ DEFINE/SYSTEM/EXECUTIVE MULTINET_FTP_ACCOUNTING_HOST lillies
$ DEFINE/SYSTEM/EXECUTIVE MULTINET_FTP_ACCOUNTING_PORT 1234
```

**Note:** The accounting port must be set to an unused port, not the port for the service on which accounting is being enabled.

The next section explains how to configure the file.

The collected accounting information can be displayed with the `MULTINET ACCOUNTING` command. See Chapter 1 of the *MultiNet Administrator's Reference* for more information about the `MULTINET ACCOUNTING` command.

## Configuration File

The Accounting configuration file is `MULTINET:ACCOUNTING.CONF`. The accounting configuration file defines:

- The port the accounting program listens on. This should be an unused port, not the port for the service on which logging is being enabled, and the same port specified to FTP or SMTP.

- The name of the file used for accounting records. This file is opened shareable and new records are always appended to it. To start a new file stop the accounting program, delete (or rename) the existing file, and restart the accounting program.
- The IP addresses of systems that are allowed to write accounting records to this host.

**Note:** After editing the configuration, stop and restart the Accounting program so that the changes can take effect.

## File Format

Follow these guidelines when entering data in the accounting configuration file:

- Allow one line for each item.
- Enter information in any order; in upper- or lowercase.
- Use a pound sign (#) or exclamation point (!) to denote comments. The accounting facility ignores all information following these characters.

The commands that can be in `MULTINET:ACCOUNTING.CONF` are:

<code>PORT <i>port_number</i></code>	The TCP port that the accounting program should listen on.
<code>PEER <i>ip-address</i></code>	The IP address of a host that is allowed to log records with the accounting software.
<code>FILENAME <i>filename</i></code>	The name of the file that the accounting records will be written to. The <code>MULTINET: device</code> is assumed if a device is not specified as part of the file specification.

## Enabling the Accounting Logger

To enable the FTP accounting logger, do the following:

```
$ MULTINET CONFIGURE/SERVER
  ENABLE ACCOUNTING
```

```
WRITE
```

```
$ @MULTINET:START_SERVER
```

## Displaying the Contents of the Logging File

To view accounting information, do the following:

```
$ MULTINET ACCOUNTING/INPUT=accounting_data_file [/output=output_file] -  
$ [/since=start_date] [/before=end_date] [/protocol={SMTP, FTP, MAIL}]  
[/CSV]
```

- *accounting\_data\_file* is the name of the logging file you want to see.
- *output filename* is the name of the file you want to call this information. If this field is omitted, the information displays to the terminal screen.
- *start\_date* is the beginning date you want the command to start with. The date format is [DD-*MMM*-YYYY [:]] [hh:mm:ss]cc. If not specified, all records display up to the end of the data found.
- The time is always in local time.
- *end\_date* is the ending date you want the command to end with. The date format is [DD-*MMM*-YYYY [:]] [hh:mm:ss]cc. If not specified, all records display until the end of the file.
- *protocol* is any combination of SMTP, FTP, or MAIL.
- CSV is the Comma Separated Values, for input to products like Excel.

## Accounting File Record Format

The accounting file is written using OpenVMS RMS records. The format of these records is defined in MULTINET\_ROOT:[MULTINET.EXAMPLES]ACCOUNTING.H, and listed below:

```
/*  
 * PDU format  
 */  
struct accountingPDU {  
    char version;  
    char type;          /* type of record */  
/*  
 * FTP:  
 *     C - Client  
 *     S - Server  
 *  
 * SMTP:  
 *     N - Network delivery (send)
```

```

*      L - Local delivery (received)
*      F - Forwarded
*      R - Returned
*      D - Delivery Receipt
*      Q - ReQueued
*
*/
char flags;          /* not currently used */
char reserved;      /* for future use */
int  payload_length; /* length (in bytes) of data after header */
int  port;          /* IP port of reporting service - 25 SMTP, 21 - FTP */
int  reporterIP;    /* IP address of reporter */
};

struct FTPaccounting_data {
    struct accountingPDU header;
    int  start_time[2]; /* VMS time that session started */
    int  end_time[2];   /* VMS time that session ended */
    int  datasent;      /* KBytes of file data sent */
    int  datarecv;      /* KBytes of file data received */
    int  filesent;      /* Number of files sent */
    int  filesrecv;     /* Number of files received */
    int  partnerIP;     /* IP address of partner */
    char user[12];      /* username that operations were done under */
};

struct SMTPaccounting_data {
    struct accountingPDU header;
    int  date[2];      /* Time of activity */
    int  msg_size;     /* size of message in bytes */
    int  from_str_size; /* size of From: string */
    int  to_str_size;  /* size of To: string */
    char from_to_str[1]; /* text of From & To string */
};

#define accounting_Close 1

typedef struct accounting_peer_info {
    struct accounting_peer_info *next;
    ulong ia;
} accounting_peer_info;

#define MAX_STRING_LEN 255

```

# FTP and IPv6

The Network Service Monitoring and Session Accounting have not yet been updated for IPv6. The same logicals and command files are used for both FTP over IPv4 and FTP over IPv6. When IPv6 is in use FTP uses the `EPSV` and `EPRT` commands. Other than the differences noted, FTP over IPv6 should be the same as FTP over IPv4.

# 16. DHCP Client

## Introduction

This chapter describes the Dynamic Host Configuration Protocol (DHCP) client.

## General Description

The DHCP client resides on the client host and dynamically sets the network configuration. The MultiNet DHCP client communicates with a DHCP server to get an IPv4 address and other configuration information. It uses this information to configure the network parameters of the host and to start up the network.

When the network starts on the host, the DHCP client communicates dynamically and automatically with the DHCP server in case reconfiguration is needed. The configuration information the client uses is defined by the policy stored in the DHCP server.

For more general DHCP information, see RFC2132 and RFC2131. Also, see Chapter 21 of this guide for general DHCP process and MultiNet DHCP server information.

MultiNet provides support for a DHCP client. Because it supports a single network interface on the host, you can only use the DHCP client to configure a single network line in MultiNet.

Process Software recommends that you do not use the DHCP client with other MultiNet components that usually need a static IPv4 address on the same host, such as DHCP server, authoritative DNS server, and GateD.

## Setting DHCP Client Parameters

The DHCP client service uses the parameters listed in the below table.

Parameters	Description
------------	-------------

CONFIGFILE	The name of the configuration file. The default is <code>MULTINET : DHCLIENT . CONF</code> .
DEBUG	<p>A decimal integer that is a bitmask of debugging levels used to select messages to pass to OPCOM and the debug log file specified in the <code>DEBUG-FILE</code> parameter. The debugging levels are (in decimal):</p> <ul style="list-style-type: none"> <li>1 Fatal Errors</li> <li>3 Errors and Warnings</li> <li>7 Informationals</li> <li>15 Debug Messages</li> <li>31 Dump Packets (formatted)</li> <li>63 Dump Packets (hex)</li> </ul> <p>By default, Fatal Errors, Errors, and Warnings are logged.</p>
DEBUG-FILE	The name of the debug log file. Use this parameter to capture debug logging in a file. The default is that debug logging is not written to any file if <code>LOG-TO-OPCOM</code> is 1. If <code>LOG-TO-OPCOM</code> is 0 (zero), the default file is <code>MULTINET : DHCLIENT . LOG</code>
INTERFACE-ROUTE	A decimal integer 1 (one). Use this parameter to add a static IP route to the MultiNet Kernel routing table. The optional <code>INTERFACE</code> keyword forces the routing to be for a locally connected interface. If the parameter is not set, the default is to create a gateway route.
LEASEFILE	The name of the file DHCP uses to save client lease information to survive across reboots. To completely clear the lease information, delete the file specified and create an empty version. The default is <code>MULTINET : DHCLIENT . DB</code> . This is not used if DHCP Safe-failover is used.
LOG-DATE	This parameter determines whether the date is included in each entry in the debug log file. The default is 0 (zero); the date is not included.
LOG-TO-OPCOM	This parameter determines whether debug logging messages are written to OPCOM. Debug messages are also written to <code>DEBUG-FILE</code> parameter value if

DEBUG-FILE is specified or this parameter is 0 (zero). Severe errors and warnings are always logged to OPCOM. The default is 1, everything is logged to OPCOM.

## Setting Up the DHCP Client

Before setting up a DHCP client, you should talk to your network administrator. The administrator may want to assign a host name to your DHCP client

If this is your first time using the DHCP client on the host, you need to do the following:

```
$ COPY MULTINET:DHCLIENT_CONF.DEFAULT MULTINET:DHCLIENT.CONF
```

Then, edit the file MULTINET:DHCLIENT.CONF to replace this line:

```
#Send host-name "testing";
```

with this line:

```
Send host-name "any hostname you want";
```

You can configure your local host to use the DHCP client when you run the MultiNet configuration utility SERVER-CONFIG.

To enable the DHCP client service and to set a parameter, use:

```
$ MULTINET CONFIGURE /SERVER
```

After configuring the local host to use the DHCP client, you can run @MULTINET:START\_SERVER to start MultiNet. You can use the same method to disable the DHCP client on the host. Here are two examples:

```
$ MULTINET CONFIGURE /SERVER
```

```
MultiNet Server Configuration Utility v5.6
```

```
[Reading in configuration from MULTINET:SERVICES.MASTER_SERVER]
```

```
SERVER-CONFIG> SELECT DHCLIENT
```

```
[The Selected SERVER entry is now DHCLIENT]
```

```
SERVER-CONFIG> SET PARAMETERS
```

```
Delete parameter "DEBUG 3"? [NO] RETURN
```

```
You can now add new parameters for DHCLIENT. An empty line terminates.
```

```
Add Parameter: DEBUG-FILE MULTINET:DHCLIENT.LOG
```

```
Add Parameter: RETURN
```

```
[Service specific parameters for DHCLIENT changed]
```

```
SERVER-CONFIG> ENABLE DHCLIENT
```

```
SERVER-CONFIG> EXIT
```

```
[Writing configuration to
```

```
MULTINET_COMMON_ROOT:[MULTINET]SERVICES.MASTER_SERVER]
```



```
[Configuration not modified, so no update needed]
```

```
$
```

This procedure creates the Network configuration data file, `MULTINET:NETWORK_DEVICES.CONFIGURATION`, as well as creating the Network Startup file, `MULTINET:START_MULTINET.COM` to reflect your system's configuration.

To activate your DHCP Client interface on a running system, use this command:

```
$ DEFINE/SYSTEM/EXECUTIVE MULTINET_DHCP_CLIENT "1"
```

After enabling the DHCP client, start the DHCP client service by restarting the MultiNet Master Server. If the DHCP client service is already running, shut it down first by issuing this command:

```
$ MULTINET_NETCONTROL_DHCLIENT_SHUTDOWN
```

Then start the MultiNet Master Server with this command:

```
$ @MULTINET:START_SERVER
```

## Disabling the DHCP Client

To disable the DHCP Client, do the following:

1. Run

```
$ MULTINET_CONFIGURE/SERVER
```

2. Issue the command:

```
SERVER-CONFIG>DISABLE_DHCLIENT
```

3. Save the configuration with the `WRITE` command and `EXIT`.
4. Reboot the system.

To avoid rebooting the system, deassign the logical `MULTINET_DHCP_CLIENT` after disabling the interface and restart the MultiNet server with the `@MULTINET:START_SERVER` command.

## DHCP Client Functions and Logicals

The DHCP Client is started as a VMS detached process (`MULTINET_DHCLIENT`) when MultiNet starts.

When the client starts, it configures the network interface (the line) with an IP address of "0.0.0.0", and then sends a DHCP discover packet to contact any DHCP server on the net. After getting an IP address and other net configuration information back from a DHCP server, it restarts the network interface with

the IP address and configures MultiNet on the host with the information it received. That information may include the default gateway, DNS domain name, host name, DNS servers' IP addresses, and other things. After the network interface is configured and started, the DHCP client goes to sleep and waits for specified events (lease expired, renewal time reached) to wake it up again for possible re-configuration.

If the DHCP client cannot get the information it needs from the DHCP server, it may re-try until it succeeds. The re-try frequency can be controlled by the configuration file.

The DHCP client process sets the following items only when configuring the network interface, if it received the appropriate information from the DHCP server:

- IP address of the network interface
- Host name of the network interface
- Domain Name
- DNS client (Resolver)
- Routes/Gateway

It may change or set the following MultiNet logical:

```
MULTINET_NAMESERVERS
```

It may change the following related OpenVMS logicals:

```
UCX$BIND_DOMAIN  
UCX$BIND_SERVER00x  
UCX$INET_HOST  
UCX$BIND_SERVER000  
UCX$INET_DOMAIN  
UCX$INET_HOSTADDR  
TCPIP$BIND_DOMAIN  
TCPIP$BIND_SERVER000  
TCPIP$BIND_SERVER00x  
TCPIP_DOMAINNAME  
TCPIP_NAMESERVERS
```

## DHCP Client Configuration

The MultiNet DHCP client uses the configuration file `MULTINET:DHCLIENT.CONF` to control the behavior of the client. Use the template file `MULTINET:DHCLIENT_CONF.DEFAULT` to start with.

To explore more configuration possibilities, read the following `dhclient.conf` descriptions. The descriptions are based on the ISC's descriptions for the UNIX version of the DHCP client configuration file. The original document can be found on the ISC's website at <http://www.isc.org/isc>.

The `dhclient.conf` file is a free-form ASCII text file. The file may contain extra tabs and new lines for formatting purposes. Keywords in the file are case-insensitive. Comments begin with the `#` character and end at the end of the line and may be placed anywhere within the file (except within quotation marks). You can use the `dhclient.conf` file to configure the behavior of the client in the following ways:

- Protocol timing
- Information requested from the server
- Information required of the server
- Defaults to use if the server does not provide certain information
- Values with which to override information provided by the server
- Values to prepend or append to information provided by the server

The configuration file can also be preloaded with addresses to use on networks that do not have DHCP servers.

## Protocol Timing

The timing behavior of the client need not be configured by the user. If no timing configuration is provided by the user, a reasonable timing behavior will be used by default — one which results in timely updates without placing an inordinate load on the server. The following statements can be used to adjust the timing behavior of the DHCP client if required.

Statement	Description
<code>backoff-cutoff time</code>	The client uses an exponential backoff algorithm with some randomness, so that if many clients try to configure themselves at the same time, they will not make their requests in lockstep. The <code>backoff-cutoff</code> statement determines the maximum amount of time the client is allowed to backoff. The default is two minutes
<code>initial-interval time</code>	The <code>initial-interval</code> statement sets the amount of time between the first attempt to reach a server and the second attempt to reach a server by recalculating the interval between messages. It is incremented by twice the current interval multiplied by a random number between

	<p>zero and one. If it is greater than the <code>backoff-cutoff</code> amount, it is set to that amount. The default is ten seconds.</p>
<code>reboot time</code>	<p>When the client is restarted, it first tries to reacquire the last address it had. This is called the <code>INIT-REBOOT</code> state. This is the quickest way to get started if it is still attached to the same network it was attached to when it last ran. The <code>reboot</code> statement sets the time that must elapse after the client first tries to reacquire its old address before it gives up and tries to discover a new address. The reboot timeout default is ten seconds.</p>
<code>retry time</code>	<p>The <code>retry</code> statement determines the time that must pass after the client has determined that there is no DHCP server present before it tries again to contact a DHCP server. By default, this is 5 minutes.</p>
<code>select-timeout time</code>	<p>It is possible to have more than one DHCP server serving any given network. It is also possible that a client may receive more than one offer in response to its initial lease discovery message. It may be that one of these offers is preferable to the other (e.g., one offer may have the address the client previously used, and the other may not). The <code>select-timeout</code> is the time after the client sends its first lease discovery request at which it stops waiting for offers from servers, assuming that it has received at least one such offer. If no offers have been received by the time the <code>select-timeout</code> has expired, the client will accept the first offer that arrives. By default, the <code>select-timeout</code> is zero seconds — that is, the client will take the first offer it sees.</p>
<code>timeout time</code>	<p>The <code>timeout</code> statement determines the amount of time that must pass between when the client begins to try to determine its address and the time it decides that it is not going to be able to contact a server. The default is 60 seconds. After the timeout has passed, if there are any static leases defined in the configuration file, or any leases remaining in the lease database that have not yet expired, the client loops through these leases attempting to validate them. If it finds one that appears to be valid, it uses that lease's address. If there are no valid static leases or unexpired leases in the lease database, the client restarts the protocol after the defined retry interval.</p>

# Lease Requirements and Requests

The DHCP protocol allows the client to request the server to send it specific information. The protocol also allows the client to reject offers from servers if they do not contain information the client needs, or if the information provided is not satisfactory. There is a variety of data contained in offers that DHCP servers send to DHCP clients. The DHCP client can request any of the DHCP options, see the table below for a list of options.

Lease Option	Description
<code>request [option] [, ... option];</code>	<p>The <code>request</code> statement causes the client to request that any server responding to the client send the client its values for the specified options. Only the option names should be specified in the request statement, not option parameters.</p> <p>For example,</p> <pre>request subnet-mask, routers;</pre>
<code>require [option] [, ... option];</code>	<p>The <code>require</code> statement lists options that must be sent in order for an offer to be accepted. Offers that do not contain all the listed options are ignored.</p>
<code>send {[ option declaration] [, ... option declaration]}</code>	<p>The <code>send</code> statement causes the client to send the specified options to the server with the specified values. These are full option declarations as described in Chapter 21 in this guide. Options that are always sent in the DHCP protocol should not be specified here. The one exception is that the client can specify a requested-lease-time option other than the default requested lease time, which is two hours. The other obvious use for this statement is to send information to the server that allows it to differentiate between this client and other clients or kinds of clients. For example,</p>

```
send host-name "my-name";
```

## Option Modifiers

In some cases, a client may receive option data from the server that is not appropriate for that client, or may not receive information that it needs, and for which a useful default value exists. It may also receive information that is useful, but needs to be supplemented with local information. To handle these needs, these option modifiers are available.

Modifier	Description
<code>append [option declaration];</code>	Use the <code>append</code> statement if the client should use the values supplied by the server followed by a value you supply. The <code>append</code> statement can only be used for options that allow more than one value to be given. This restriction is not enforced. If you ignore it, the behavior is unpredictable.
<code>default [option declaration];</code>	Use the <code>default</code> statement to specify a default value if no value was supplied by the server.
<code>prepend [option declaration];</code>	Use the <code>prepend</code> statement if the client should use a value you supply followed by the values supplied by the server. The <code>prepend</code> statement can only be used for options that allow more than one value to be given. This restriction is not enforced. If you ignore it, the behavior is unpredictable.
<code>supersede [option declaration];</code>	Use the <code>supersede</code> statement if the client should always use a locally-configured value or values rather than whatever is supplied by the server.

# Lease Declarations

A *lease* statement consists of the `lease` keyword, followed by a left curly brace ( `{` ), followed by one or more lease declaration statements, followed by a right curly brace ( `}` ).

```
lease {lease-declaration [ ... lease-declaration ] }
```

The DHCP client may decide after some period of time (see *Protocol Timing*) that it is not going to succeed in contacting a server. At that time, it consults its own database of old leases and tests each one that has not yet timed out by pinging the listed router for that lease to see if that lease could work. It is possible to define one or more fixed leases in the client configuration file for networks where there is no DHCP or BOOTP service, so that the client can still configure its address automatically. This is done with the `lease` statement.

**Note:** The `lease` statement is also used in the `dhclient.db` file in order to record leases that have been received from DHCP servers. Some of the syntax for leases as described below is only needed in the `dhclient.db` file. Such syntax is documented here for completeness.

The following lease declarations are possible:

Declaration	Description
<code>bootp;</code>	The <code>bootp</code> statement indicates that the lease was acquired using the BOOTP protocol rather than the DHCP protocol. It is never necessary to specify this in the client configuration file. The client uses this syntax in its lease database file.
<code>filename "string";</code>	The <code>filename</code> statement specifies the name of the boot filename to use. This is not used by the standard client configuration script, but is included for completeness.
<code>fixed-address ip-address;</code>	The <code>fixed-address</code> statement sets the IP address of a particular lease. This is required for all lease statements. The IP address must be specified as a dotted quad (e.g., 12.34.56.78).

<pre>interface "string";</pre>	<p>The <code>interface</code> statement indicates the interface on which the lease is valid. If set, this lease will be tried only on a particular interface. When the client receives a lease from a server, it always records the interface number on which it received that lease. If predefined leases are specified in the <code>dhclient.conf</code> file, the interface should also be specified, although this is not required.</p>
<pre>option option-declaration;</pre>	<p>The <code>option</code> statement specifies the value of an option supplied by the server, or, in the case of predefined leases declared in <code>dhclient.conf</code>, the value that the user wants the client configuration script to use if the predefined lease is used.</p>
<pre>renew date; rebind date; expire date;</pre>	<p>The <code>renew</code> statement defines the time at which the DHCP client should begin trying to contact its server to renew a lease that it is using.</p> <p>The <code>rebind</code> statement defines the time at which the DHCP client should begin to try to contact any DHCP server in order to renew its lease.</p> <p>The <code>expire</code> statement defines the time at which the DHCP client must stop using a lease if it has not been able to contact a server in order to renew it.</p> <p>These declarations are set automatically in leases acquired by the DHCP client, but must be configured in predefined leases: a predefined lease whose expiration time has passed will not be used by the DHCP client. Dates are specified as follows:</p> <pre>weekday year/month/day hour:minute:second</pre> <pre>W YYYY/MM/DD HH:MM:SS</pre>



	<p><i>W</i> is the day of the week, from zero (Sunday) to six (Saturday).  <i>YYYY</i> is the year, including the century.  <i>MM</i> is the number of the month, from 01 to 12.  <i>DD</i> is the day of the month, counting from 01.  <i>HH</i> is the hour, from 00 to 23.  <i>MM</i> is the minute, from 00 to 59.  <i>SS</i> is the second, from 00 to 59.</p> <p>The time is always in Greenwich Mean Time, not local time.</p>
<code>server-name "string";</code>	The <code>server-name</code> statement specifies the name of the boot server name to use. This is not used by the standard client configuration script.
<code>script "script-name";</code>	The <code>script</code> statement specifies the file name of the DHCP client configuration script. This script is used by the DHCP client to set the interface's initial configuration prior to requesting an address, to test the address once it has been offered, and to set the interface's final configuration once a lease has been acquired. If no lease is acquired, the script is used to test predefined leases, if any, and also called once if no valid lease can be identified. The default value for “ <code>script-name</code> ” is “ <code>MULTINET:DHCLIENT-SCRIPT.COM</code> ”.

## Other Declarations

Declaration	Description
<code>reject ip-address;</code>	The <code>reject</code> statement causes the DHCP client to reject offers from servers who use the specified address as a server identifier. This can be used to avoid being configured by rogue or misconfigured DHCP servers,

although it should be a last resort; better to track down the bad DHCP server and fix it.

## Example

The following configuration file is used on an ALPHA machine with VMS 7.1.

```
# template of MULTINET:DHCLIENT.CONF
send host-name "lambda2";
send dhcp-lease-time 3600;
prepend domain-name-servers 127.0.0.1;
request subnet-mask, broadcast-address, time-offset, routers,
        domain-name, domain-name-servers, host-name;
require subnet-mask, domain-name-servers;
timeout 60;
retry 60;
reboot 10;
select-timeout 5;
initial-interval 2;
script "multinet:dhclient-script.com";
reject 10.10.10.10; # reject the offer from this DHCP server
```

The first line, starting with the #, is a comment line. The last line rejects the offer from the DHCP server with an IP address of 10.10.10.10. This is not a simple `dhclient.conf` file. In many cases, it is sufficient to just create an empty `dhclient.conf` file and let the DHCP client use default values.

## Troubleshooting the DHCP Client

### How do I know the DHCP client has configured my network successfully?

There are two ways to do it:

Check if the `MULTINET_DHCP_CLIENT` logical is equal to "1":

```
$ SHOW LOGICAL MULTINET DHCP CLIENT
"MULTINET_DHCP_CLIENT" = "1" (LNM$SYSTEM_TABLE)
$
```

Run the MULTINET CONFIGURE /INTERFACE command.

1. Check the line DHCP client for which the interface is enabled.
2. Check that the line DHCP Client Flag is set.

```
$ MULTINET CONFIGURE /INTERFACE
MultiNet Network Configuration Utility v5.6
[Reading in MAXIMUM configuration from MULTINET:MULTINET.EXE]
[Reading in configuration from MULTINET:NETWORK_DEVICES.CONFIGURATION]
NET-CONFIG>SHOW
Interface                               Adapter           CSR Address       Flags/Vector
-----                               -
se0 (Shared VMS Ethernet/FDDI)         -NONE-           -NONE-           -NONE-
  [TCP/IP: 10.10.10.10]
  [VMS Device: EWA0, Link Level: Ethernet]
  DHCP Client for the interface is enabled
Official Host Name:                    dumdummy.fuges.com
Domain Nameserver:                     127.0.0.1
Timezone:                               EST
Timezone Rules:                        US/EASTERN
Load UCX $QIO driver:                  TRUE
Load PWIP (Pathworks) driver:         TRUE
DHCP Client Flag:                      1
NET-CONFIG>QUIT
$
```

## What if I cannot ping an IP address on the internet?

If you can ping the same IP address from another host and the network interface has been configured by the DHCP client, check the gateway and route configuration on the host.

## What if I can ping a host by its IP address but not by its name?

- The DNS client on the host may not be configured right. Type

```
$ SHOW LOGICAL MULTINET NAMESERVERS
to make sure the DNS client information is correct.
```

- The DNS server may be down.

# Why is the local address "0.0.0.0" when I use "\$ multinet configure /interface" and then use "show"?

The DHCP client has failed to allocate an IP address. The possible reasons and solutions are:

Reason	Solution
There is no DHCP server on the net.	Set up a DHCP server.
The DHCP server is not configured correctly.	Modify the DHCP server configuration.
The DHCP client is configured to reject the DHCP server.	Reconfigure the DHCP client to not reject the DHCP server.
The hostname selection process failed.	Use another host name.
There are no IP addresses available in the DHCP server.	Increase the IP address on the DHCP pool server.

## Where can I find the status information of the DHCP client?

- The file `MULTINET:DHCLIENT-SCRIPT-ENV.TMP` contains the most recent environment variables used by the DHCP client script file to configure the network.
- The file `MULTINET:DHCLIENT.DB` contains the DHCP client lease history.
- The file `MULTINET:DHCLIENT.LOG` contains information about the DHCP client process.

The `MULTINET:DHCLIENT.LOG` file is not created by the default setting of the DHCP client. To create this log file, configure the DHCP client to enable the error and debug logging:

```
$ MULTINET CONFIGURE /SERVER
SERVER-CONFIG>SELECT DHCLIENT
SERVER-CONFIG>SET PARAMETERS
Add Parameter:DEBUG 7
Add Parameter:RETURN
SERVER-CONFIG>WRITE
SERVER-CONFIG>RESTART
```

# 17. Configuring the Font Server

This chapter explains how to use the MultiNet font server to provide fonts for X11R5 (and later) X servers on your network. To understand the material in this chapter, you should be familiar with font administration on X11R5 servers.

## Understanding the Font Server

The MultiNet font server makes fonts on your OpenVMS system available to remote X11R5 (and later) X servers without using a distributed file system, such as NFS, or file transfer via FTP or TFTP.

The main advantages of font servers over distributed file systems or file transfer are:

- Simplicity of font administration.
- Redundancy. X servers can use multiple font servers. If one font server fails or is unavailable, the X server can request fonts from another font server.

You can add font servers to an X server font search path the same way you add directories to the font search path. For example, you can add a font server to the font search path of X servers running on UNIX systems with the `xset +fp` and `xset fp+` commands.

- When an X server needs a font, it sends a request to the font server.
- If the requested font is on the font server, the font is transferred to the X server.
- If the font is not on the font server, the X server continues to search the rest of the font search path, which may include other font servers.

You can also configure the font server to return the names of other font servers (known as *alternates*) that the X server can search when the font server fails to find a requested font.

# The Font Server Configuration File

The MultiNet font server obtains its configuration parameters from the configuration file `MULTINET:FONT_SERVER.CONFIGURATION`, which is equivalent to the `/usr/lib/X11/fs/config` configuration file used by font servers on UNIX systems. Although the file names are different, the file formats are identical; you can use configuration files from UNIX systems on your OpenVMS host.

The configuration file is an ASCII text file that contains a list of configuration parameter names and values. Each parameter name is followed by an equals sign (=) and the desired value.

The table below describes the font server configuration parameters.

Parameter	Accepted Values	Description
<code>cache size</code>	cardinal number	Specifies the size of the font server's font cache. To improve font access speed, specify a large font cache. Default: 10000.
<code>catalogue</code>	list of strings	Lists font path element names, delimited by commas. The MultiNet font server supports only a single catalogue ( <i>all</i> ), which contains all specified fonts.
<code>alternate-server</code>	list of strings	Lists alternate servers for this font server.
<code>client-limit</code>	cardinal number	Specifies the number of clients this font server supports before refusing service.
<code>default-point-size</code>	cardinal number	Specifies the default point size (in decipoints) for fonts that don't specify a point size.
<code>default-resolutions</code>	comma-delimited list of integers	Specifies the default resolutions the server supports. This information may be used as a hint for pre-rendering, and substituted for scaled fonts which do not specify a resolution.
<code>error-file</code>	string	Specifies the log file into which all warnings and errors are written.

port	cardinal number	Specifies the TCP port on which the server listens for connections.
trusted-clients	comma-delimited list of host names	Determines which hosts can use the font server. If you don't specify any hosts (the default), all hosts can use the font server. If you specify hosts, they are the only ones that can use the font server.

An example font server configuration file follows.

```
# Font server configuration file
# MultiNet Font Server
# Specify the font directories to export.
#
# WARNING: The file DECW$FONT_DIRECTORY.DAT must exist.  If it does not, you
# can create it with the command $ MU FONT MKFONTDIR [directory,...]
#
catalogue = sys$common:[sysfont.decw.100dpi],
           sys$common:[sysfont.decw.75dpi],
           sys$common:[sysfont.decw.common],
           sys$common:[sysfont.decw.cursor16],
           sys$common:[sysfont.decw.cursor32]
#
# Uncomment this line to start logging errors to a file on disk.
# Restart the font server to put logging into effect.
#
#error-file = MultiNet:fs.errors
#
# in decipoints
default-point-size = 120
default-resolutions = 75,75,100,100
```

## Specifying Font Servers

All X11R5 (and later) servers use the same syntax for specifying font servers:

```
transport/host_name:port_number[/catalogue]
```

- *transport* is "TCP."
- *host\_name* is the name of the host on which the font server is running.
- *port\_number* is the port on which the MultiNet font server listens for requests from remote X servers. By convention, the MultiNet font server listens on port 7000.

- *catalogue* is the catalogue the MultiNet font server provides (by default, "all"). Catalogues are the equivalent of search paths on the font server. For details on defining catalogues, see the *Defining Font Catalogues* section.

## Supported Font Types

The MultiNet font server supports the following font formats:

BDF	MIT SNF	Speedo
DECwindows	PCF	

MultiNet also includes two commands for converting font formats:

MULTINET FONT COMPILE	Compiles BDF fonts into PCF format. Type <code>HELP MULTINET FONT COMPILE</code> for online help.
MULTINET FONT UNCOMPILE	Converts fonts supported by the font server into BDF format. Type <code>HELP MULTINET FONT UNCOMPILE</code> for online help.

The default font alias file name is `DECW$FONT_ALIAS.DAT` to match the default font values used by OpenVMS.

## Enabling the Font Server

To enable the MultiNet font server, use the `SERVER-CONFIG` utility (`MULTINET CONFIGURE /SERVER`). For example, to enable the font server on a single OpenVMS system, enter:

```
$ MULTINET CONFIGURE /SERVER
SERVER-CONFIG>ENABLE FONTSERVER
SERVER-CONFIG>RESTART
Configuration modified, do you want to save it ? [YES] YES
```



```
SERVER-CONFIG>QUIT  
$
```

# Getting Information About the Font Server

This section describes how to get information about a specific font server. Use the `MULTINET FONT` command to obtain the following information about the font server:

- Current font server configuration (see the *Checking the Font Server Configuration* section)
- Names of available fonts (see the *Listing Available Fonts* section)
- Font file data (see the *Viewing Font Data* section)

## Checking the Font Server Configuration

To check the status of the MultiNet font server, enter:

```
$ MULTINET FONT INFO /SERVER=font_server_name:port_number
```

- *font\_server\_name* is the name of the font server you want to check. Use the font server name syntax described in the *Specifying Font Servers* section.
- *port\_number* is the port on which the font server listens. By convention, the MultiNet font server listens on port 7000.

The following example shows the information generated by this command on a system that acts as a font server with no alternates.

```
$ MULTINET FONT INFO /SERVER=WHORFIN:7000  
name of server: WHORFIN:7000  
version number: 2  
vendor string: ABC, Incorporated  
vendor release number: 5001  
maximum request size: 16384 longwords (65536 bytes)  
number of catalogues: 1  
all  
Number of alternate servers: 0  
number of extensions: 0  
$
```

For more information, type `HELP MULTINET FONT INFO`.

## Listing Available Fonts

To list the names of available fonts on a font server, enter:

```
$ MULTINET FONT LIST /SERVER=font_server_name font_spec
```

- *font\_server\_name* is the name of the font server from which you want to obtain the list of fonts. Use the font server name syntax described in *Specifying Font Servers*.
- *font\_spec* is a font specification, in which you may include wildcard characters.

The following example shows the command that lists all "fixed" fonts on the font server.

```
$ MULTINET FONT LIST /SERVER=WHOREFIN:7000 *FIXED*
```

```
-misc-fixed-bold-r-normal--0-0-75-75-c-0-iso8859-1
-misc-fixed-bold-r-normal--13-120-75-75-c-70-iso8859-1
-misc-fixed-bold-r-normal--13-120-75-75-c-80-iso8859-1
-misc-fixed-bold-r-normal--15-140-75-75-c-90-iso8859-1
-misc-fixed-bold-r-semicondensed--0-0-75-75-c-0-iso8859-1
-misc-fixed-bold-r-semicondensed--13-120-75-75-c-60-iso8859-1
-misc-fixed-medium-r-normal--0-0-75-75-c-0-iso8859-1
-misc-fixed-medium-r-normal--10-100-75-75-c-60-iso8859-1
-misc-fixed-medium-r-normal--13-120-75-75-c-70-iso8859-1
-misc-fixed-medium-r-normal--13-120-75-75-c-80-iso8859-1
-misc-fixed-medium-r-normal--14-130-75-75-c-70-iso8859-1
-misc-fixed-medium-r-normal--15-140-75-75-c-90-iso8859-1
-misc-fixed-medium-r-normal--20-200-75-75-c-100-iso8859-1
-misc-fixed-medium-r-normal--8-80-75-75-c-50-iso8859-1
-misc-fixed-medium-r-normal--9-90-75-75-c-60-iso8859-1
-misc-fixed-medium-r-semicondensed--0-0-75-75-c-0-iso8859-1
-misc-fixed-medium-r-semicondensed--12-110-75-75-c-60-iso8859-1
-misc-fixed-medium-r-semicondensed--13-120-75-75-c-60-iso8859-1
-sony-fixed-medium-r-normal--0-0-100-100-c-0-iso8859-1
-sony-fixed-medium-r-normal--16-120-100-100-c-80-iso8859-1
-sony-fixed-medium-r-normal--24-170-100-100-c-120-iso8859-1
$
```

For more information, type `HELP MULTINET FONT LIST`.

## Viewing Font Data

To list the data that comprises a specific font, log into the host running the font server, and enter:

```
$ MULTINET FONT SHOW /SERVER=font_server_name font_spec
```

- *font\_server\_name* is the name of the font server from which you want to obtain the font data. Use the font server name syntax described in the *Specifying Font Servers* section.
- *font\_spec* is a font specification, in which you may include wildcard characters.

For information on other MULTINET FONT SHOW qualifiers, refer to the *MultiNet Administrator's Reference*.

The following example shows the command that lists the data for two characters in the Courier font:

```
$ MULTINET FONT SHOW /SERVER=WHOREIN:7000 /START=52 /END=53 *COURIER*
opened font *COURIER*
Direction: Left to Right
Range: 32 to 255
Default char: 32
Min bounds:
Left: -2      Right: 1      Ascent: -1      Descent: -5      Width: 6
Max bounds:
Left: 3       Right: 7       Ascent: 9       Descent: 2       Width: 6
Font Ascent: 8  Font Descent: 2
FONT      -Adobe-Courier-Bold-O-Normal--11-80-100-100-M-60-ISO8859-1
FOUNDRY Adobe
FAMILY_NAME      Courier
WEIGHT_NAME      Bold
SLANT            0
SETWIDTH_NAME    Normal
ADD_STYLE_NAME
PIXEL_SIZE       11
POINT_SIZE       80
RESOLUTION_X     100
RESOLUTION_Y     100
SPACING M
AVERAGE_WIDTH   60
CHARSET_REGISTRY      ISO8859
CHARSET_ENCODING     1
CAP_HEIGHT        6
X_HEIGHT          5
FACE_NAME         Courier Bold Oblique
COPYRIGHT         Copyright (c) 1984, 1987 Adobe Systems Incorporated. All
Rights Reserved. Copyright (c) 1988, 1991 Digital Equipment Corporation.
All Rights Reserved.
NOTICE           No mark
_DEC_DEVICE_FONTNAMES PS=Courier-BoldOblique
_DEC_PRODUCTINFO   DECwindows Fonts V2.2, 07-Nov-1991
RELATIVE_SETWIDTH  50
RELATIVE_WEIGHT    70
CHARSET_COLLECTIONS ASCII ISO8859-1 ADOBE-STANDARD
FULL_NAME         Courier Bold Oblique
RESOLUTION        138
QUAD_WIDTH        6
char #52 '4'
Left: 0          Right: 5          Ascent: 7          Descent: 0          Width: 6
---##
--###
-#-##
#--#-
```

```
#####  
--##-  
--##-  
char #53 '5'  
Left: 0      Right: 6      Ascent: 7      Descent: 0      Width: 6  
--####  
-##---  
-###--  
--##-  
---##-  
#--##-  
###---  
$
```

For more information, type `HELP MULTINET FONT SHOW`.

# Controlling the MultiNet Font Server

This section describes how to control the font server.

Use the `MULTINET NETCONTROL FONTSERVER` command for the following tasks:

- Starting the font server (see the *Starting the Font Server* section)
- Stopping the font server (see the *Stopping the Font Server* section)
- Restarting the font server (see the *Restarting the Font Server* section)
- Reloading the font server configuration (see the *Reloading the Font Server Configuration* section)
- Flushing the font server cache (see the *Flushing the Font Server Cache* section)
- Resetting the font server (see the *Resetting the Font Server* section)

## Starting the Font Server

To start the MultiNet font server, enter:

```
$ MULTINET NETCONTROL FONTSERVER START  
< FS Server Started, process id pid
```

When the font server starts, it reads the master configuration file, `MULTINET:FONT_SERVER.CONFIGURATION`. For information about the master configuration file, see *The Font Server Configuration File* section.

# Stopping the FS Server

To stop the MultiNet font server, enter:

```
$ MULTINET NETCONTROL FONTSERVER SHUTDOWN  
< FS Server Shutdown
```

# Restarting the Font Server

Restarting the font server is a convenient alternative to first stopping and then starting it as described in the *Starting the Font Server* section and the *Stopping the Font Server* section.

When the font server restarts, it reads the configuration file, `MULTINET:FONT_SERVER.CONFIGURATION`. For information about the font server configuration file, see *The Font Server Configuration File* section.

To restart the MultiNet font server, enter:

```
$ MULTINET NETCONTROL FONTSERVER RESTART  
< FS Server Started, process id pid
```

**Note:** Because the font server provides fonts on request, restarting does not disrupt any connections.

# Reloading the Font Server Configuration

Changes to the font server configuration file (`MULTINET:FONT_SERVER.CONFIGURATION`) only take effect when the font server is started, restarted, or when the configuration files are reloaded. Reloading the font server allows you to reload font server configuration files without restarting the font server.

For information on the font server configuration file, see *The Font Server Configuration File* section. To reload the font server configuration file, enter:

```
$ MULTINET NETCONTROL FONTSERVER RELOAD  
< WHORFIN.EXAMPLE.COM Network Control 5.6 at Wed 26-Nov-2019 1:33PM-PST  
< OK: FS server configuration reloading
```

**Note:** Because the font server provides fonts on request, reloading does not disrupt active connections.

## Flushing the Font Server Cache

To improve performance, the font server keeps copies of requested fonts in a cache. To flush the font cache, enter:

```
$ MULTINET NETCONTROL FONTSERVER FLUSH
< WHORFIN.EXAMPLE.COM Network Control 5.6 at Wed 26-Nov-2019 1:36PM-PST
< OK: Font Server cache flushed
```

The size of this cache is defined in the font server configuration file, `MULTINET:FONT_SERVER.CONFIGURATION`. For details about the font server configuration, see *The Font Server Configuration File* section.

## Resetting the Font Server

For convenience, MultiNet provides a `RESET` command to flush and reload the font server. To reset the font server, enter:

```
$ MULTINET NETCONTROL FONTSERVER RESET
< WHORFIN.EXAMPLE.COM Network Control 5.6 at Wed 26-Nov-2019 1:37PM-PST
< OK: Font Server reset
```

## Defining Font Catalogues

Font catalogues are the font server equivalent of X server *font search paths*. To make fonts available via the font server, add the directories in which they reside to the "catalogue" line in the font server configuration file `MULTINET:FONT_SERVER.CONFIGURATION`.

For example, the default catalogue definition supplied with MultiNet is defined as:

```
catalogue = sys$common:[sysfont.decw.100dpi],
            sys$common:[sysfont.decw.75dpi],
            sys$common:[sysfont.decw.common],
```

```
sys$common:[sysfont.decw.cursor16],  
sys$common:[sysfont.decw.cursor32]
```

If you modify the font server configuration file, the changes only take effect when you start, restart, reload, or reset the font server.

## Adding Fonts to the Font Server

To make a new font available via the font server:

1. Install the font file in the appropriate font directory on the font server host. If the font is in BDF format, you may want to convert the font into PCF format with the `MULTINET FONT COMPILE` command to improve font server performance (type `HELP MULTINET FONT COMPILE` for online help).
2. Update the font directory's `DECW$FONT_DIRECTORY.DAT` file with the `MULTINET FONT MKFONTDIR` command (type `HELP MULTINET FONT MKFONTDIR` for online help). This command creates the `DECW$FONT_DIRECTORY.DAT` file.

**Note:** If the `DECW$FONT_DIRECTORY.DAT` file is not found, the font server fails. Be sure to run `MULTINET FONT MKFONTDIR` *manually* in each DECwindows font directory in which you add a font file. Failing to do so may result in the font server not serving the standard DECwindows fonts.

**Note:** When using `MU FONT MKFONTDIR` you must specify directories, not logical disks. For example, `MU FONT MKFONTDIR MULTINET:` is not valid.

3. If desired, add an alias for the new font to the font directory's `DECW$FONT_ALIAS.DAT` file.
4. Make sure the font directory is included in the "catalogue" statement in the font server configuration file `MULTINET:FONT_SERVER.CONFIGURATION`. For details, see the *Defining Font Catalogs*

section. If you must modify the configuration file, reload the font server configuration (see the *Reloading the Font Server Configuration* section).

For example, to configure the MultiNet font server to provide the fonts included in the NCDware 3.0 distribution for VMS, include the following font directories in your catalogue definition:

```
NCD_ROOT: [FONTS.PCF.100DPI]
NCD_ROOT: [FONTS.PCF.75DPI]
NCD_ROOT: [FONTS.PCF.DW100DPI]
NCD_ROOT: [FONTS.PCF.DW75DPI]
NCD_ROOT: [FONTS.PCF.MISC]
NCD_ROOT: [FONTS.PCF.XOL]
```



# 18. Configuring Remote Systems with RARP, BOOTP, and DHCP Server

This chapter explains how to configure MultiNet to supply network configuration data to remote client systems when they boot.

MultiNet provides three services that provide configuration data to remote systems:

- RARP (Reverse Address Resolution Protocol)
- BOOTP (Bootstrap Protocol) (responds only to BOOTP clients)
- DHCP (Dynamic Host Configuration Protocol) (responds to both BOOTP and DHCP clients)

The BOOTP and DHCP servers allow a network administrator to configure various hosts on the network from a single location. In addition to the management of IP addresses, BOOTP and DHCP also provide configuration parameters to clients, such as default gateway, domain name server, and subnet mask.

## Choosing a Network Configuration Server

This section presents a brief description of the services and some criteria for deciding which protocol and services to use. The following table lists the advantages and disadvantages of the three protocols:

Service	Advantages and Disadvantages
RARP	Supplies IP addresses only.
BOOTP	Lets you provide vendor-specific configuration data. Works in conjunction with TFTP. Provides static configuration data only.

DHCP	Lets you provide vendor-specific configuration data. Provides both BOOTP and DHCP services. Provides dynamic configuration for mobile computing, but does not solve all problems of mobile computing.
------	---

## RARP (Reverse Address Resolution Protocol)

RARP's sole function is to provide IP addresses to hosts that broadcast RARP requests with their hardware addresses.

## BOOTP (Bootstrap Protocol)

BOOTP sends IP addresses and other configuration data to hosts that broadcast BOOTP requests. Because some BOOTP clients require more data to boot than can fit in a BOOTP response, BOOTP provides a means for specifying the location of a boot file. The BOOTP client can then load the file using TFTP (Trivial File Transfer Protocol). Usually, the data in the boot file (such as an X server for an X terminal) is specific to the vendor of the BOOTP client software.

The BOOTP service responds to BOOTP requests only. If you are using the BOOTP-only service, DHCP services are not available. The BOOTP server is provided for backwards compatibility for those sites not wanting to change their configuration.

## DHCP (Dynamic Host Configuration Protocol)

DHCP is an extension of the BOOTP protocol. DHCP sends IP addresses and other configuration data to hosts that broadcast DHCP requests. DHCP "leases" an IP address to a remote system for a finite time. DHCP lets you manage IP addresses and configuration data for a "pool" of remote systems, which makes DHCP useful for mobile computers that connect to multiple subnets.

As with BOOTP, DHCP provides a means for specifying the location of a boot file which the DHCP client can load using TFTP. For details on creating a downloadable boot file for a specific type of host, refer to the vendor's documentation.

If you are using the DHCP server, all BOOTP services are available as well. A MultiNet host can have only one of the servers (BOOTP or DHCP) enabled because both use the same port.

**Note:** Your BOOTP and DHCP configuration files must be located in the default location of MULTINET : in order for an automatic conversion to occur.

## Using RARP

RARP (Reverse Address Resolution Protocol) is commonly used by diskless hosts to determine their Internet address. While ARP (Address Resolution Protocol) lets hosts resolve Internet addresses into Ethernet addresses, RARP lets them resolve Ethernet addresses into Internet addresses. Configuring the MultiNet RARP server consists of:

1. Obtaining the data needed by each RARP client (see the *Obtaining Data for RARP Clients* section).
2. On HP Ethernet interfaces only, enabling RARP packet reception (see the *Enabling RARP Packet Reception on HP Ethernet Interface* section).
3. Enabling and starting RARP (see the *Enabling and Starting RARP Service* section).
4. Adding client systems to the RARP configuration file (see the *Adding Clients to the RARP Configuration File* section).
5. Reloading the RARP configuration (see the *Reloading RARP Configuration* section).

**Note:** Because RARP clients send their requests in a link-layer broadcast (Ethernet, for example) and most routers do not forward link-layer broadcasts, make sure the MultiNet system and all its RARP clients are on the same physical network.

## Obtaining Data for RARP Clients

Obtain the IP and Ethernet addresses for each client you want to use with RARP. To obtain a client's Ethernet interface address, refer to the interface documentation.

Ethernet addresses are expressed as six hexadecimal numbers (ranging from 0 to ff) separated by colons. IP addresses are expressed in dotted-decimal format.

## Enabling RARP Packet Reception on HP Ethernet Interfaces

If your MultiNet system has an HP Ethernet interface, enable RARP packet reception with the `MULTINET SET /INTERFACE` command. For example, to enable RARP packet reception on the `se0` interface:

```
$ MULTINET SET/INTERFACE/VMS_DEVICE=XQA0:/LINK_LEVEL=ETHERNET/RARP SE0
```

To automatically enable RARP packet reception when MultiNet starts, make sure to add the `/RARP` qualifier to the `MULTINET SET /INTERFACE` command line in the custom initialization command procedure for the interface, as described in Chapter 11.

## Enabling and Starting RARP Service

To enable RARP, use `SERVER-CONFIG`:

```
$ MULTINET CONFIGURE /SERVER  
MultiNet Server Configuration Utility 5.6  
[Reading in configuration from MULTINET:SERVICES.MASTER_SERVER]  
SERVER-CONFIG>ENABLE RARP  
SERVER-CONFIG>EXIT  
[Writing configuration to  
MULTINET_COMMON_ROOT:[MULTINET]SERVICES.MASTER_SERVER]  
$
```

Once you have enabled RARP, start it by restarting the MultiNet Master Server:

```
$ @MULTINET:START_SERVER
```

# Adding Clients to the RARP Configuration File

The MultiNet RARP server uses Ethernet-to-IP address translations from the `MULTINET:RARP.CONFIGURATION` file. Each single-line entry in `RARP.CONFIGURATION` contains an Ethernet address and the corresponding Internet address.

The following `RARP.CONFIGURATION` sample shows the IP addresses assigned to the hosts with Ethernet addresses `aa:00:04:00:45:12` and `00:0c:00:17:12:67`.

```
#
#   This is a sample RARP database. It provides the mapping between
#   ethernet addresses and IP addresses as used by RARP.
#
# ethernet address      ip address
# -----
aa:00:04:00:45:12     192.0.0.1
00:0c:00:17:12:67     192.0.0.2
```

## Reloading RARP Configuration

After modifying the `MULTINET:RARP.CONFIGURATION` file, reload the RARP configuration with the following command:

```
$ MULTINET NETCONTROL RARP RELOAD
```

## Using BOOTP

The MultiNet BOOTP (Bootstrap Protocol) service lets your OpenVMS system help diskless hosts and other network devices establish network connectivity. The remote system broadcasts a BOOTP request over the network with its Ethernet address. The BOOTP server looks up the host's address in a configuration file (`MULTINET:BOOTP-SERVER.CONFIGURATION`) and responds with the host's IP address, subnet mask, gateway address, initial load file, and any other data needed by the client. Using this information, the client can boot from the network.

Starting with MultiNet V3.5, MultiNet includes two BOOTP servers: An older server provided for backwards compatibility for those sites not wanting to change their configuration, and a newer DHCP/BOOTP server that provides features not present in the older, BOOTP-only server.

For details on BOOTP see RFC-951, "Bootstrap Protocol," and RFC-1084, "BOOTP Vendor Information Extensions."

Configuring the BOOTP server involves:

1. Obtaining the data required by each BOOTP client (see *Obtaining Data for BOOTP Clients*).
2. Enabling and starting BOOTP (see *Enabling and Starting BOOTP*).
3. Modifying the BOOTP configuration file (see *Modifying the BOOTP Configuration File*).
4. Reloading the BOOTP configuration (see *Reloading the BOOTP Configuration*).
5. Disabling debug messages, if desired (see *Disabling BOOTP OPCOM Messages*).

**Note:** While BOOTP is often used with clients and servers on the same network, they can be on different physical networks. Most routers can be configured to forward BOOTP requests; refer to your router documentation.

## Obtaining Data for BOOTP Clients

Make a list of the configuration parameters (known as BOOTP options) required by the devices you want to configure using BOOTP. The table below lists BOOTP options.

Because some network devices require large amounts of information or vendor-specific configuration at boot time, BOOTP lets you specify the path names of additional configuration files the client can download from TFTP servers. For details on creating downloadable configuration files for a specific host, refer to the vendor's documentation.

**Note:** If you are running DNS, make sure you use the same IP address and host name data used by your primary site's DNS servers. If you are using host tables instead of DNS, make sure you use the same IP address and host name data listed in `MULTINET:HOSTS.LOCAL`.

## Enabling and Starting BOOTP

You can enable BOOTP with `SERVER-CONFIG`:

```
$ MULTINET CONFIGURE /SERVER  
MultiNet Server Configuration Utility 5.6  
[Reading in configuration from MULTINET:SERVICES.MASTER_SERVER]  
SERVER-CONFIG>ENABLE BOOTP
```

```
SERVER-CONFIG>EXIT
[Writing configuration to
MULTINET_COMMON_ROOT:[MULTINET]SERVICES.MASTER_SERVER]
$
```

**Note:** BOOTP cannot run while DHCP is enabled because both services use the same port. You can use SERVER-CONFIG to disable DHCP.

After enabling BOOTP, start it by restarting the MultiNet Master Server:

```
$ @MULTINET : START_SERVER
```

## Modifying the BOOTP Configuration File

MultiNet supplies a MULTINET:BOOTP-SERVER.CONFIGURATION file that contains comments and a number of examples to help you enter information for your hosts.

## BOOTP Options for the BOOTP Server

This table describes the options you can define for each host and an example of each option.

Field	Description	Example
bf	File downloaded by TFTP to the client at boot time. This file is supplied by the device vendor. The file must exist and be world-readable. If the file is not found, a null file specification is returned.	bf="mom\$load:xncd16_1t"
bs	Bootfile size. If the value is the string "auto" or no value is given, the server automatically determines the size. Otherwise, the specified value is passed verbatim. The size is expressed in 512-byte blocks.	bs=auto or bs=24

cs	Space-separated list of "quote of the day" server IP addresses. The cookie (as in "fortune cookie") server is described in RFC-865.	cs=192.41.228.92
ds	Space-separated list of domain name server IP addresses	ds=192.41.228.65
gw	IP address of the default gateway	gw=128.2.13.1
ha	Hardware address of the client. The format of the hardware address depends on the hardware type (ht). Specify the hardware type (ht) before the hardware address (ha).	ha=00DD00C88900
hd	Home directory for the boot files	hd="sys\$sysroot: [bootp-boot files]"
hn	Flag requesting the host name to be sent to the client. When an entry contains this tag, the contents of the name field (the initial string of characters on each record up to, but not including the first colon) are sent to the client. If the name field is greater than 58 characters, only the host field (up to the first period) is sent. If the host field by itself does not fit, no value is sent.	hn
ht	Hardware address type. The hardware type must be interpreted before the hardware address (ha). Valid values are the hardware type, expressed as a decimal number as defined by the RFCs or a text string that maps to the hardware type number:  <pre> ethernet 1    ieee803    6    chaos  5 ethernet3 2    tr         6    arcnet 7 ether     1    token-ring 6    ax.25  3 ether3    2    pronet     4 </pre>	ht=ethernet, ht=6
im	Space-separated list of Imagen-type "Impress" server IP addresses	im=192.168.228.92 192.168.228.93
ip	IP address of the host	ip=192.168.228.82



lg	Space-separated list of MIT-LCS UDP log server IP addresses	lg=192.168.228.42
lp	Space-separated list of LPR server IP addresses	lp=192.168.228.37
ns	Space-separated list of IEN-116 name server IP addresses	ns=192.168.228.77
rl	Space-separated list of RLP (Resource Location Protocol) server IP addresses	rl=192.168.228.19
sa	IP address of a boot server	sa=192.168.228.222
sm	Subnetwork mask	sm=255.255.255.192
tc	Template host label. Use the <code>tc</code> field to "include" information from another entry in the configuration file. You may create a common entry for a group of hosts, such as a specific vendor's X terminals. Use the <code>tc</code> field to combine information specific to each model. Information in the current entry overrides information included by the <code>tc</code> field. A <code>tc</code> entry may also "include" another entry with a <code>tc</code> field of its own.	tc=global.dummy
td	TFTP directory. Used to reference part of a directory that may be hidden from the client via the TFTP server.	td="TFTP\$DIR:"
to	Time offset (in seconds) east of GMT for the client. <b>Error! Reference source not found.</b> lists accepted values. BOOTP uses negative numbers west of GMT and positive numbers east of GMT. See the table below for the time offset values you can specify in this field.	to=25200
ts	Space-separated list of time server IP addresses	ts=192.41.228.77
Tn	"Generic" tag of the type "Tn=value," <ul style="list-style-type: none"> <li>n is the number assigned the option.</li> </ul>	T123="Hello World" or T124=FFFE2CEF

	<ul style="list-style-type: none"> <li>• <code>value</code> is either ASCII data enclosed in quotes or binary data expressed as hexadecimal digits.</li> </ul> <p>When expressing binary data that represents short or long values, be sure to check the byte order to compensate for the difference between OpenVMS byte order and network byte order. For values with known tags, the server can convert between the two. For values in generic tags, however, the server cannot tell the difference between a four-byte binary string and an unsigned long value.</p>	
<code>vm</code>	"Vendor magic" to send: "auto", "rfc1048", "rfc1084", "cmu", or a dotted-decimal value. Vendor magic is always "rfc1084" when using DHCP. Default: auto.	<code>vm="rfc1048"</code>

This table provides time offset values you can specify in the `to` field:

Timezone	Time Offset	DST Time Offset	Timezone	Time Offset	DST Time Offset
AST/ADT	-14400	-10800	MET/MET-DST	7200	3600
BST	0	3600	MST/MDT	-25200	-21600
CET/CET-DST	7200	3600	NST/NDT	-12600	-9000
CST/CDT	-21600	-18000	NZST	86400	90000
EET/EET-DST	10800	14400	PST/PDT	-28800	-25200
EST/EDT	-18000	-14400	SST	+28800	none
GMT	0	none	UTC	0	none
HST	-36000	none	WET/WET-DST	3600	7200
JST	32400	none	YST/YDT	-32400	-28800

# Guidelines for the BOOTP Configuration File

The following guidelines govern modification of the `MULTINET:BOOTP-SERVER.CONFIGURATION` file:

- Edit the configuration file with any text editor.
- Use a pound sign (#) in the first column of the line to designate a comment line. Comment and blank lines are ignored by the server.
- Specify the hardware type (`ht`) before the hardware address (`ha`).
- Specify IP addresses in dotted-decimal notation.

**Note:** If you enter an IP address with leading zeros as part of the address (for example, 192.41.012.011), the octets with leading zeros are interpreted as octal values rather than decimal values.

- For readability, limit each entry to one line when possible. Otherwise, put each field on a separate line.
- Separate entry fields with a colon (:). When lines are continued on another line, separate fields with a colon followed by a backslash. You should start each new line with a tab followed by a colon. Here are examples of the two different entry styles:

```
ncd16s:\
    :ht=ethernet:\
    :bf="mom$load:xncd16_lt":\
    :gw=10.41.228.71:\
    :sm=255.255.255.192:\
    :ds=10.41.228.65:\
    :to=25200:
tree:tc=ncd16s:ha=0000C0545F24:ip=10.41.228.75:
```

- Use the `tc` field as an "include" statement to succinctly provide additional information for an individual device, as shown in the example entries above. The entry called "tree" is for an individual NCD terminal. Including the `tc` option adds all of the information in the "ncd16s" entry to the "tree" entry.

The `tc` field lets you create a common entry for a class of hosts (such as a vendor's X terminals) that conveys generic information. Entries that include `tc` options supply information specific to an individual terminal, such as its IP address.

Information in the individual entry overrides the information included by the `tc` field.

- When specifying more than one server for the `cs`, `ds`, `im`, `lg`, `lp`, `ns`, `rl` and `ts` fields, separate subsequent server values with spaces.

## Using a UNIX bootptab File

If you are also running a BOOTP server on a UNIX system, you can use the UNIX system's `bootptab` configuration file after making the following changes:

- Copy the `bootptab` file to `MULTINET:BOOTP-SERVER.CONFIGURATION`.
- Change the syntax of directories and file names to OpenVMS format.
- Do not add names that conflict with existing entries.

## Reloading the BOOTP Configuration

After modifying `MULTINET:BOOTP-SERVER.CONFIGURATION`, reload the BOOTP configuration with the following command:

```
$ MULTINET NETCONTROL BOOTP RELOAD
```

## Disabling BOOTP OPCOM Messages

After you test your BOOTP configuration, you may want to suppress some of the messages the BOOTP server sends to OPCOM by changing the debug level of the BOOTP server, as shown in this example:

```
$ MULTINET NETCONTROL BOOTP DEBUG -1
```

If you want this change to take place each time MultiNet is started, use the `SERVER-CONFIG SET PARAMETERS` command as follows:

```
$ MULTINET CONFIGURE /SERVER
```

```
MultiNet Server Configuration Utility 5.6  
[Reading in configuration from MULTINET:SERVICES.MASTER_SERVER]  
SERVER-CONFIG> SELECT BOOTP
```

```
[The Selected SERVER entry is now BOOTP]
SERVER-CONFIG>SET PARAMETERS
Delete parameter "bootfile MULTINET:BOOTP-SERVER.CONFIGURATION" ? [NO] RETURN
You can now add new parameters for BOOTP.  An empty line terminates.
Add Parameter: debug -1
Add Parameter: RETURN
[Service specific parameters for BOOTP changed]
SERVER-CONFIG>RESTART
Configuration modified, do you want to save it first ? [YES] RETURN
[Writing configuration to MULTINET_COMMON_ROOT:[MULTINET]
SERVICES.MASTER_SERVER]
%RUN-S-PROC_ID, identification of created process is 20600046
SERVER-CONFIG>EXIT
[Configuration not modified, so no update needed]
$
```

## Using DHCP

The MultiNet DHCP (Dynamic Host Configuration Protocol) server lets your OpenVMS system help diskless hosts and other network devices establish network connectivity. The DHCP server provides all of the functions of BOOTP plus dynamic addressing and additional configuration options.

The DHCP server offers a network host a temporary lease of an IP address rather than an ownership of an IP address, such as BOOTP does. The lease identifies the length of time the client can safely use its assigned IP address. The network administrator sets the lease length using parameters in the configuration file. It is recommended that the network administrator assign lease lengths based on the number of network users and the number of available IP addresses the DHCP server can assign. To configure the DHCP server:

1. Obtain the data required by each DHCP client (see *Obtaining Data for DHCP Clients*).
2. Modify the DHCP configuration file (see *Introducing the Configuration File*).
3. Enable and start the DHCP server (see *Enabling and Starting DHCP*).
4. If you modify the configuration file after starting the DHCP server, reload the DHCP server (see *Reloading the DHCP Configuration*).

**Note:** DHCP uses DNS for host names and IP addresses; thus, a malfunction in your DNS server can affect the DHCP server.

# DHCP Process

DHCP goes through an initializing, selecting, requesting, binding, renewal, rebinding, and expiration cycle when negotiating for an IP address, as shown in the diagram below. The process is as follows:

1. The client just added or relocated on the network requests an IP address by broadcasting a DHCPDISCOVER message to the local subnet over the well-known BOOTP server port. (The client can also go through a BOOTP router or relay agent to forward the DHCPDISCOVER to additional remote DHCP servers.) This is the initializing state.
2. The participating DHCP servers respond with a DHCPOFFER message if they have a valid configuration for the client. The client may get many of these messages, which contain the IP address and configuration data. (The servers make sure to reserve the addresses so as not to accidentally offer them to another client.) At this point the client enters the selecting state.
3. After selecting an address, the client broadcasts the selected address and name of the "winning" server (Server 1 in the below figure) using a DHCPREQUEST message. This is the requesting state. All the other servers can now safely unreserve their addresses.
4. Server 1 sends the client a DHCPACK (acknowledgement) message with the negotiated IP address, the lease, and the network configuration parameters. The client now enters the binding state and can fully use the assigned IP address.
5. About halfway through the lease, the client sends Server 1 another DHCPREQUEST for a lease renewal, and enters the renewal state. If the server deems the lease renewable, it sends back another DHCPACK to update the lease (including any new parameters). The client now returns to the binding state, as in step 4.
6. If the client cannot renew the lease (such as if Server 1 is down), the client waits until about 87.5% of the way through the lease and broadcasts another DHCPREQUEST to all DHCP servers. Any server can now return a DHCPACK containing the extended lease and updated parameters. This is the rebinding state.

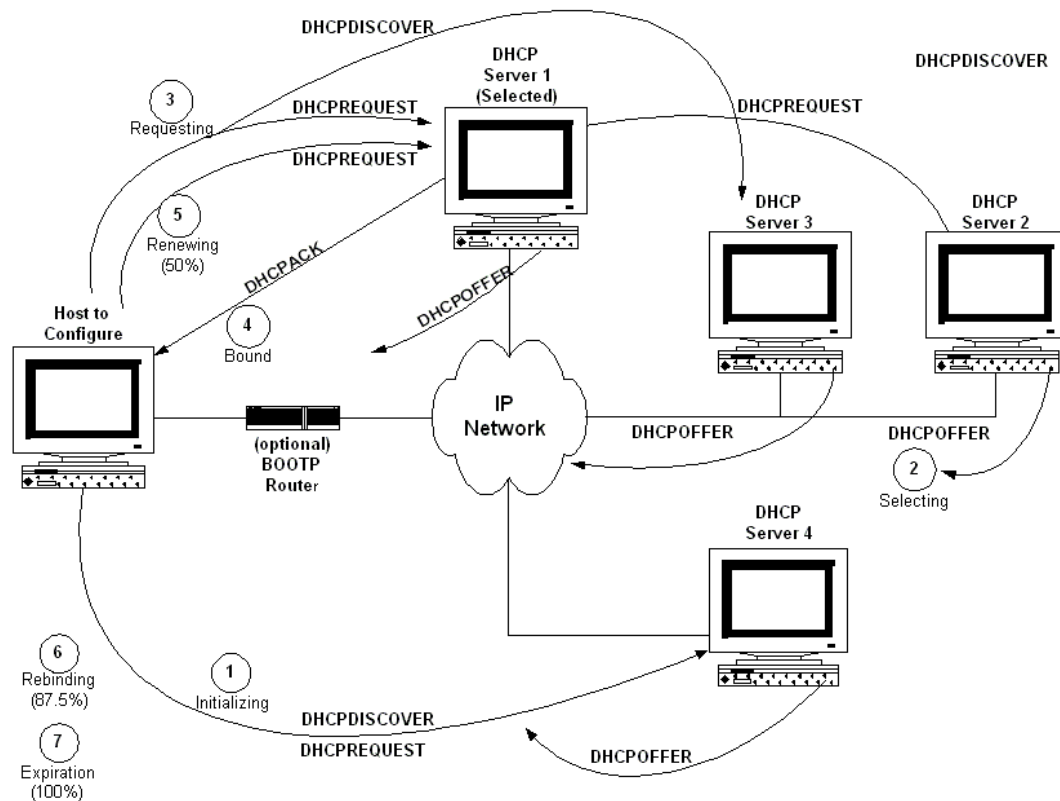
- When the lease reaches 100% expired, or a server sends back a DHCPNAK negative acknowledgement message, the client must give up the IP address. It then returns to the initializing state and has to start the address negotiation over again.

See the DHCP RFCs for more information. DHCP is defined in RFC 2131 and RFC 2132.

Two DHCP servers are recommended for a network. The benefit of having more than one server is if one fails another is available to continue processing requests, ensuring that all hosts (old and new) are serviced continuously. Refer to *DHCP Safe-failover Introduction* for more information.

## Obtaining Data for DHCP Clients

Make a list of the configuration parameters (known as DHCP options) required by the devices you want to configure using DHCP.



# Enabling and Starting DHCP

You can enable the DHCP server with `SERVER-CONFIG`:

```
$ MULTINET CONFIGURE /SERVER
MultiNet Server Configuration Utility 5.6
[Reading in configuration from MULTINET:SERVICES.MASTER_SERVER]
SERVER-CONFIG>ENABLE DHCP
SERVER-CONFIG>EXIT
[Writing configuration to
MULTINET_COMMON_ROOT:[MULTINET]SERVICES.MASTER_SERVER]
$
```

**Note:** DHCP cannot run while the BOOTP server is enabled because both servers use the same port. Because the DHCP server provides BOOTP service as well, there is no need to run the BOOTP service.

After you have enabled DHCP, start the DHCP server by restarting the MultiNet Master Server. If the DHCP server is already running, shut it down first.

```
$ MULTINET NETCONTROL DHCP SHUTDOWN
$ @MULTINET:START_SERVER
```

# Checking the DHCP Configuration

After modifying the configuration file, it is good practice to verify the syntax by running the DHCP server interactively specifying the `-t` flag, as follows:

```
$ dhcpd ::= $multinet:dhcpd.exe
$ dhcpd -t [-cf <config-file>]
```

You can test both the configuration file and the lease file using the `-T` flag:

```
$ dhcpd "-T" [-cf <config-file>] [-lf <lease-file>]
```

The `-t` flag causes the DHCP server to run just far enough to read and parse the configuration file. The DHCP server displays a copyright notice as well as a message for each syntax error encountered. If the DHCP server displays only the copyright notice, the configuration file has no syntax errors.

The `-T` flag causes the DHCP server to run just far enough to read and parse the configuration and lease files.



DHCPD can be made to use an alternate configuration file with the `-cf` flag, or an alternate lease file with the `-lf` flag. If you do not specify the `-cf` flag, the DHCP server reads the default configuration file `MULTINET:DHCPD.CONF`. If you do not specify the `-lf` flag, the DHCP server reads the default lease file `MULTINET:DHCPD.LEASES`. Because of the importance of using the same lease database at all times when running DHCPD in production, these flags should be used **only** for testing lease files or database files in a non-production environment.

## Reloading the DHCP Configuration

If you modify `MULTINET:DHCPD.CONF` after starting the DHCP server, restart DHCP with the following command so the DHCP server rereads `DHCPD.CONF`:

```
$ MULTINET NETCONTROL DHCP RESTART
```

## DHCP Conversion Tool

The `DHCP_CONVERSION_TOOL` assists in moving from the DHCP server in MultiNet V4.1 and earlier to the DHCP server in MultiNet V5.2 and later. This tool converts the configuration files and state file from the 4.1 DHCP server to the format of the configuration and lease files of the 5.5 DHCP server. It is run automatically by the command procedure that starts the MultiNet Master Server, `MULTINET:START_SERVER.COM`. However, it is recommended that customers run the conversion tool and double check the output ahead of time.

The conversion tool reads the old BOOTP and DHCP configuration files and writes out a configuration file in the new format, representing a merging of the two old configuration files, with DHCP configuration information being preferred. All information from the old configuration files is in the new file. Information that was duplicated or that does not have an equivalent in the new configuration file is represented with comment lines.

The conversion tool then reads the old state file and writes out a lease file in the new format.

The conversion tool preserves the old configuration and state files where they were. The network administrator can use them to validate the new configuration and lease files.

The conversion tool may also be run directly. The names of the input and output files may be specified on the command line. If they are not specified, the tool will prompt for them. Enter “NONE” for the file name if you do not have the input file. For example, if you do not have a BOOTP configuration file:

```
$ dhcpconvert := $multinet:dhcp_conversion_tool.exe
$ dhcpconvert dhcp-server.configuration NONE -
_$ dhcp-state.dat dhcpd.conf dhcpd.leases
```

Enter “?” as the sole parameter to get a help message. The default file names for the five files are:

MULTINET:DHCP-SERVER.CONFIGURATION	the old DHCP configuration file
MULTINET:BOOTP-SERVER.CONFIGURATION	the old BOOTP configuration file
MULTINET:DHCPD.CONF	the new DHCP configuration file
MULTINET:DHCP-STATE.DAT	the old DHCP state file containing the lease status
MULTINET:DHCPD.LEASES	the new DHCP lease file

The conversion tool may produce large configuration files. To reduce the size of these configuration files, set the following logical name before running the conversion tool:

```
$ define MULTINET_DHCP_CONV_BRIEF YES
```

## Introducing the Configuration File

MultiNet supplies a `MULTINET:DHCPD.CONF` file that contains comments and a number of examples to help you enter information for your hosts. You can edit the configuration file with any text editor. Add or remove entries as needed.

The `dhcpd.conf` file is a free-form ASCII text file. The file may contain extra tabs and newlines for formatting purposes. Keywords in the file are case-insensitive. Comments may be placed anywhere within the file (except within quotation marks). Comments begin with the `#` character and end at the end of the line. See the sample `DHCP.CONF` file at the end of this chapter.

The file consists of a list of statements. Statements fall into two categories: *parameters* and *declarations*.

*Parameter* statements always say one of the following:

- How to do something (e.g., how long a lease to offer).
- Whether to do something (e.g., should the DHCP server provide addresses to unknown clients).
- What parameters to provide to the client (e.g., use gateway 220.177.244.7).

Global parameters are at the beginning of the file. Some examples of global parameters are the organization's domain name and the addresses of the name servers (if they are common to the entire organization).

It is legal to specify host addresses in parameters as domain names rather than as numeric IP addresses. If a given hostname resolves to more than one IP address (for example, if that host has two ethernet interfaces), both addresses are supplied to the client.

Both the `shared-network` statement and the `subnet` statement can have parameters.

The most obvious reason for having subnet-specific parameters is that each subnet, of necessity, has its own router. For example, something like:

```
option routers 204.254.239.1;
```

**Note:** The address here is specified numerically. This is not required. If you have a different domain name for each interface on your router, it is perfectly appropriate to use the domain name for that interface instead of the numeric address. However, there may be only one domain name for all of a router's IP addresses, and it would not be appropriate to use that name here.

Parameters starting with the `option` keyword correspond to actual DHCP options. Parameters that do not start with the `option` keyword either control the behavior of the DHCP server (e.g., how long a lease the DHCP server will give out), or specify client parameters that are not optional in the DHCP protocol (for example, `server-name` and `filename`).

Each host can have host-specific parameters. These could include such things as the:

- Hostname option.
- Name of a file to upload (the `filename` parameter).
- Address of the server from which to upload the file (the `next-server` parameter).

In general, any parameter can appear anywhere that parameters are allowed, and will be applied according to the scope in which the parameter appears.

All parameters must be specified first before you can specify any declarations that depend on those parameters. Parameters should be set inside declarations so they can be set on a per-subnet or a per-host basis.

*Declarations* are used to:

- Describe the topology of the network.
- Describe clients on the network.
- Provide addresses that can be assigned to clients.
- Apply a group of parameters to a group of declarations.

Declarations about network topology include the `subnet` and the `shared-network` declarations.

For every subnet to be served, and for every subnet connected to the DHCP server, there must be one `subnet` declaration. This declaration tells the DHCP server how to recognize that an address is on that particular subnet. A `subnet` declaration is required for each subnet even if no addresses will be dynamically allocated on that subnet.

There are different declarations required for different situations. The following is a list of the basic declarations in a configuration file.

- For clients with dynamically assigned addresses, a `range` declaration must appear within the `subnet` declaration, or a `pool` declaration.
- For clients with statically assigned addresses, or for installations where only known clients will be served, each client must have a `host` declaration.
- If parameters are to be applied to a group of declarations that are not related strictly on a per subnet, class, or pool basis, the `group` declaration can be used.

Some installations have physical networks allowing more than one IP subnet to operate. For example, if your site has a requirement that 8-bit subnet masks be used, but a department with a single physical ethernet network expands beyond 254 nodes, you may have to run two 8-bit subnets on the same ethernet until a new physical network is added. In this case, you can enclose the `subnet` declarations for these two networks in a `shared-network` declaration.

Some sites may have departments that have clients on more than one subnet, but it may be desirable to offer those clients a uniform set of parameters that are different than what would be offered to clients from other departments on the same subnet.

- For clients declared explicitly with `host` declarations, enclose these declarations in a `group` declaration using the parameters that are common to that department.
- For clients with dynamically assigned addresses, one way to group parameter assignments is by network topology. Alternately, `host` declarations can provide parameters and if they have no fixed-address parameter, the clients get an address dynamically assigned. See the `host` declarations example below.
- Clients can be grouped into `classes` and assigned IP addresses from specific `pools`.

When a client is to be booted, its boot parameters are determined by consulting the following *scopes* in this order:

1. Client's `host` declaration (if any).
2. Group declaration (if any) that enclosed the host declaration.
3. Subclass declaration for the subclass the client belongs to (if any).
4. Class declaration for the class the client belongs to (if any).
5. Pool declaration that the assigned IP address comes from (if any).
6. Subnet declaration for the subnet on which the client is booting.
7. Shared-network declaration (if any) containing that subnet.
8. Top-level parameters that may be specified outside of any declaration.

When searching for a `host` declaration, the DHCP server looks for one with a `fixed-address` parameter that matches the subnet or shared network on which the client is booting.

Imagine that you have a site with a lot of NCD X-Terminals. These terminals come in a variety of models, and you want to specify the boot files for each model. One way to do this would be to have `host` declarations for each server and group them by model:

```
group {
    filename "Xncd19r";
    next-server ncd-booter;
    host ncd1 { hardware ethernet 0:c0:c3:49:2b:57; }
    host ncd4 { hardware ethernet 0:c0:c3:80:fc:32; }
    host ncd8 { hardware ethernet 0:c0:c3:22:46:81; }
}
group {
    filename "Xncd19c";
    next-server ncd-booter;
    host ncd2 { hardware ethernet 0:c0:c3:88:2d:81; }
    host ncd3 { hardware ethernet 0:c0:c3:00:14:11; }
}
group {
    filename "XncdHMX";
    next-server ncd-booter;
    host ncd1 { hardware ethernet 0:c0:c3:11:90:23; }
    host ncd4 { hardware ethernet 0:c0:c3:91:a7:8; }
    host ncd8 { hardware ethernet 0:c0:c3:cc:a:8f; }
}
```

## Address Allocation

Address allocation is done when a client is in the `INIT` state and has sent a `DHCPDISCOVER` message. When the DHCP server is looking for an IP address to allocate to a client, it checks first

- if the client has an active lease on an IP address, or
- if the client has an expired lease on an IP address that has not been reassigned.

It then follows these rules:

- If a lease was found but the client is not permitted to use it, then the lease is freed (if it was not expired already).
- If no lease is found or a lease was found and the client is not permitted to use the address, then the server looks for an address that is not in use and that the client is permitted to have among the list of address pools on the client's subnet.
- If no addresses are found that can be assigned to the client, then no response is sent to the client.
- If an address is found that the client is permitted to have, then the address is allocated to the client.

**Note:** IP addresses that have never been assigned are chosen over those that have previously been assigned to other clients.

If the client thinks it has a valid lease and sends a `DHCPREQUEST` to initiate or renew that lease, the server has three choices. It can

- Ignore the `DHCPREQUEST`.
- Send a `DHCPNAK`, telling the client to stop using the address.
- Send a `DHCPACK`, telling the client to use the address.

If the server finds the requested address and that address is available to the client, the server sends a `DHCPACK`.

If the address is no longer available or the client is not permitted to have it, the server sends a `DHCPNAK`.

If the server knows nothing about the address, the server remains silent. However, if the address is incorrect for the network segment to which the client is attached and the server is authoritative for that segment, the server sends a `DHCPNAK`.

## Address Pools

`Pool` declarations let you have different allocation policies for different address allocation pools. A client may be denied access to one pool, but allowed access to another pool on the same network segment.

A `pool` declaration is used to specify how a group of addresses should be treated differently than another group of addresses, even if they are on the same network segment or subnet.

For example, you can provide a large set of addresses assigned to DHCP clients that are known to your DHCP server, while at the same time providing a small set of addresses that are available for unknown clients. If you have a firewall, you can arrange for addresses from one pool to have access to the Internet, while addresses in another pool do not have access to the Internet. The following example illustrates how you could set up a pair of `pool` declarations.

```
subnet 10.0.0.0 netmask 255.255.255.0 {
  option routers 10.0.0.254;
  # Unknown clients get this pool.
  pool {
    option domain-name-servers bogus.example.com;
    max-lease-time 300;
    range 10.0.0.200 10.0.0.253;
    allow unknown clients;
  }
  # Known clients get this pool.
  pool {
    option domain-name-servers ns1.example.com, ns2.example.com;
    max-lease-time 28800;
    range 10.0.0.5 10.0.0.199;
    deny unknown clients;
  }
}
```

You can also set up entirely different subnets for known and unknown clients. This is possible because address pools exist at the level of shared networks, so address ranges within pool declarations can be on different subnets, as long as they are on the same shared network.

## Pool Permit Lists

The above example shows that address pools can have permit lists. A permit list controls which clients are allowed access to the address pool and which clients are not allowed access. Each entry in a permit list is introduced with the `allow` or `deny` keyword. The following table describes the four possibilities for eligibility to addresses from the address pool.

If a pool has...	Then...
a permit list	only those clients that match specific entries on the permit list are eligible for addresses from the pool.

a deny list	only those clients that do not match any entries on the deny list are eligible for addresses from the pool.
both a permit list and a deny list	only clients that match the permit list and do not match the deny list are eligible for addresses from the pool.
neither a permit list nor a deny list	all clients are eligible for addresses from the pool.

Range declarations that appear outside of *pool* declarations in the same shared-network are grouped into two pools: one which allows all clients for *range* statements with the `dynamic-bootp` keyword and one which denies dynamic bootp clients for *range* statements without the `dynamic-bootp` keyword.

As described in the *Address Allocation* section, the DHCP server checks each IP address to see if the client is permitted to use it, in response to both `DHCPDISCOVER` and `DHCPREQUEST` messages. The DHCP server checks both the address pool permit lists and the relevant in-scope `allow` and `deny` statements.

See the table of DHCP statements for the recognized `allow` and `deny` statements. They can be used to permit or refuse access to known or unknown clients, members of a class, dynamic bootp clients, or all clients.

## Client Classing

You can separate clients into classes, treating each client differently depending on what class it is in. To separate clients into classes, use conditional statements (see the *Conditional Behavior* section) or a `match` statement within a `class` declaration. You can specify a limit on the total number of clients within a particular class or subclass that may hold leases at one time using the `lease limit` statement. You can specify automatic subclassing based on the contents of the client packet using the `spawn with` statement.

To add clients to classes based on conditional evaluation, write a conditional statement to match the clients you want in the class. Then, put an `add` statement in the conditional's list of statements. For example, to identify requests coming from Microsoft Windows RAS servers:

```
if substring (option dhcp-client-identifier, 1, 3) = "RAS" {
    add "ras-clients";
}
```



An equivalent way to do this is to specify the conditional expression as a matching expression in the class statement. For example:

```
class "ras-clients" {  
    match if substring (option dhcp-client-identifier, 1, 3) = "RAS";  
}
```

**Note:** Whether you use matching expressions or add statements (or both) to classify clients, you must write a class declaration for any class that you use.

If you want no match statement and no in-scope statements for a class, the declaration looks like this, for example:

```
class "ras-clients" {  
}
```

**Important!** The add statement adds the client to the class **after** the address assignment has been completed. This means the client will not be affected by pool permits related to that class if the client is a member of a class due to an add statement.

## Subclasses

In addition to classes, you can declare subclasses. A subclass is a class having the same name as a regular class but with a specific submatch expression that is hashed for quick matching. It is quicker to find five subclasses within one class than it is to find five classes with match expressions. The following example illustrates how to code for subclasses:

```
class "allocation-class-1" {  
    match hardware;  
}  
class "allocation-class-2" {  
    match hardware;  
}  
subclass "allocation-class-1" 1:0:0:c4:aa:29:44;  
subclass "allocation-class-1" 1:8:0:2b:4c:39:ad;  
subclass "allocation-class-2" 1:8:0:2b:a9:cc:e3;
```

```
subnet 10.0.0.0 netmask 255.255.255.0 {
  pool {
    allow members of "allocation-class-1";
    range 10.0.0.11 10.0.0.50;
  }
  pool {
    allow members of "allocation-class-2";
    range 10.0.0.51 10.0.0.100;
  }
}
```

The data following the class name in the `subclass` declaration is a constant value used in matching the match expression for the class. During class matching, the server evaluates the match expression and looks up the result in the hash table. If a match is found, the client is considered a member of both the class and the subclass.

You can specify subclasses with or without scope (i.e., statements). In the above example, the sole purpose of the subclass is to allow some clients access to one address pool, while other clients are given access to the other pool. Thus, these subclasses are declared without any statements (scope). If you wanted to define different parameter values for some clients, you would declare those subclasses with scopes.

For example: if you had a single client needing some configuration parameters, while most did not, you might write the following `subclass` declaration for that client:

```
subclass "allocation-class-2" 1:08:00:2b:a1:11:31 {
  option root-path "samsara:/var/diskless/alphapc";
  filename "/tftpboot/netbsd.alphapc-diskless";
}
```

In the previous examples, subclassing is being used as a way to control address allocation on a per-client basis. However, it is possible to use subclassing in ways that are not specific to clients. For example, to use the value of the `vendor-class-identifier` option to determine what values to send in the `vendor-encapsulated-options` option. See the *Vendor Encapsulated Options* section.

**Note:** If you are using `match hardware`, the hardware address is preceded by the hardware type. In this example, the “1:” indicates Ethernet.

# Per-Class Limits on Dynamic Address Allocation

The number of clients in a class that can be assigned leases can be limited. This limiting makes it difficult for a new client in a class to get an address. Once a class has reached its limit, the only way a new client in that class can get a lease is for an existing client to relinquish its lease, either by

- letting it expire, or
- sending a DHCPRELEASE packet.

The following example illustrates how to specify classes with lease limits.

```
class "limited-1" {  
    lease limit 4;  
}
```

This produces a class in which a maximum of four members may hold leases at one time.

If you want to provide clients at a particular site with more than one IP address, but do not want to provide these clients with their own subnet, nor give them an unlimited number of IP addresses from the network segment to which they are connected, you can create a spawning class and use lease limits. A spawning class is a class that produces subclasses automatically based on what the client sends.

Many cable modem head-end systems can be configured to add a Relay Agent Information option to DHCP packets when relaying them to the DHCP server. These systems typically add a circuit ID or remote ID option that uniquely identifies the customer site. The following example illustrates how to write a class declaration to take advantage of these relay agent options to create lease limited classes on the fly:

```
class "customer" {  
    match if exists agent.circuit-id;  
    spawn with option agent.circuit-id;  
    lease limit 4;  
}
```

With this class declaration, whenever a request comes in from a customer site, the circuit ID option is checked against the class's hash table.

- If a subclass matches the circuit ID, the client is classified in that subclass.
- If no subclass matches the circuit ID, a new subclass is created and logged in the `dhcpcd.leases` file and the client is classified in the new subclass.

Once a client is classified, it is treated according to the rules of the class; as in the example above, being subjected to the per-site limit of four leases.

**Note:** The use of the subclass spawning mechanism is not restricted to relay agent options. This particular example is given only because it is a straightforward one.

## Conditional Behavior

The DHCP server can be configured to perform conditional behavior depending on the packets it receives.

Conditional behavior is specified using the `if` statement and the `else` or `elsif` statements. A conditional statement can appear anywhere that a regular statement can appear, and can enclose one or more such statements. The following is an example of a conditional statement.

```
if option dhcp-user-class = "accounting" {
  max-lease-time 17600;
  option domain-name "accounting.example.org";
  option domain-name-servers ns1.accounting.example.org,
                             ns2.accounting.example.org;
} elsif option dhcp-user-class = "engineering" {
  max-lease-time 17600;
  option domain-name "engineering.example.org";
  option domain-name-servers ns1.engineering.example.org,
                             ns2.engineering.example.org;
} else {
  max-lease-time 600;
  option domain-name "misc.example.org";
  option domain-name-servers ns1.misc.example.org,
                             ns2.misc.example.org;
}
```

Both the `if` statement and the `elsif` continuation statement take expressions that, when evaluated, produce a boolean result. See the *Expressions* section for more information.

- If the expression evaluates to true, then the statements enclosed in braces following the `if` statement are executed. All subsequent `elsif` and `else` clauses are skipped.
- If the expression evaluates to false, then the statements enclosed in braces following the `if` statement are not executed and each subsequent `elsif` clause is checked until an `elsif` clause is encountered that evaluates to true.
- If such an `elsif` clause is found, then the statements in braces following it are executed. Any subsequent `elsif` and `else` clauses are skipped.

- If all the `if` and `elsif` clauses are checked but none of their expressions evaluate to true, then if there is an `else` clause, then the statements enclosed in braces following the `else` clause are evaluated.

**Note:** Boolean expressions that evaluate to null are treated as false in conditionals.

## DNS Dynamic Updates Within DHCP

The DHCP server performs dynamic updates to DNS using DNS's dynamic updating functionality. To be sure that updates are allowed from the DHCP server, see Chapter 10, Host Tables and DNS. The `allow-update { address_match_list };` statement in the `Zone` section enables the DNS server to allow updates from that system.

The following statements in the DHCP server's configuration file are related to dynamic updating:

```
allow/deny dynamic-update;  
allow/deny update-A-record;  
allow/deny name-by-client;  
invalid-ddns-chars {fail | discard | replace ["chars"]};
```

Dynamic updates can be enabled or disabled by using the `allow/deny dynamic-update` statement in the configuration file. The default is to not perform dynamic updates. Dynamic updates can be turned on or off on a per subnet basis.

**Note:** Dynamic updates are not done at all for static assignments to BOOTP clients, and the support for static assignments to DHCP clients is to add DNS entries only.

When dynamic updating is enabled, the DHCP server determines the client's Fully Qualified Domain Name (FQDN) and assigns it an IP address. The FQDN is determined either by what the client sends or by what is in the configuration file. This behavior is controlled by the `allow/deny name-by-client` statement in the configuration file.

If you use the `deny name-by-client` statement or if the client does not send a name, you must specify the host name in the configuration file using one of the following methods:

- Using option `host-name` “name” (see the *Host Name Generation* section)
- Specifying `use-host-decl-names` on in conjunction with host declarations.

If the hostname specified by the client contains invalid characters for DNS, the DHCP server can handle them one of three ways:

- Consider it a failure and not do the dynamic update.
- Throw away the invalid characters.
- Replace the invalid characters with specified valid characters.

This behavior is controlled by the `invalid-ddns-chars` statement in the configuration file.

The FQDN and IP address are used in the dynamic update to create a PTR resource record (RR). The DHCP server also optionally creates an A RR. This option is enabled or disabled by using the `allow/deny update-A-record` statement in the configuration file. The default is to not create the A RR. This can be set on a per subnet basis. See Chapter 10, Host Tables and DNS, the *DNS Zone Information Files* section for information about PTR resource records and A resource records.

When dynamic updating is allowed, the DHCP server adds the resource records whenever an IP address is leased to a client. The RRs are deleted if the IP address is released or if the IP address is leased to a different client. Also, the RRs are deleted for expired leases periodically.

## Transaction Signatures (TSIG)

The DHCP server supports using Transaction Signatures (TSIG) on dynamic updates to DNS. Note that you need a DNS server that supports TSIG, such as MultiNet’s BIND server.

The use of TSIG can be enabled or disabled by using the `secure-ddns` statement in the configuration file. The default is to not use TSIG. The use of TSIG can be turned on or off on a per subnet basis. Turn on the use of TSIG using:

```
secure-ddns on;
```

For each DNS server that you want to use TSIG with, you must specify a key using the `key` declaration:

```
key ip-address {
[ algorithm "hmac-md5"; ]
key-id "key-name";
secret "key";
}
```

- `ip-address` is the IP address of the DNS server
- `algorithm` specifies the algorithm to use. The only supported algorithm is "hmac-md5". This statement is optional.
- `key-id` specifies the name of the key as a string. This must match the key name being used by the DNS server (e.g., configured in `named.conf`).
- `secret` specifies the secret key to use in base-64 format. This must match the secret key used by the DNS server (in `named.conf`).

An example `key` declaration for the DNS server at IP address 10.9.8.7 is:

```
key 10.9.8.7 {  
key-id "dhcp-tsig";  
secret "A5vhC+DjsocELGEYhj0iBBSQRgJvxnY/emD0C3kRtEpo";  
};
```

## Host Name Generation

Some DHCP clients require that the server send them a host name. The MultiNet DHCP server can generate a host name if it cannot get the host name in another way. The generated host name can contain parts of the host's IP address, client ID, and/or MAC address. This host name is sent to the client and is combined with the domain name to create the Fully Qualified Domain Name (FQDN) required for dynamic DNS updates. See the *DNS Dynamic Updates Within DHCP* section. As described in the *DNS Dynamic Updates Within DHCP* section, the `allow/deny name-by-client` statement in the configuration file controls whether the DHCP server uses information from the client to determine the host name and FQDN.

The DHCP server generates a host name if it is enabled to do so and either

- `allow name-by-client` is specified and the client does not send a host name, or
- `deny name-by-client` is specified and the DHCP server does not find a host name in the configuration file or in DNS (if `get-lease-hostnames` is set).

To enable the DHCP server to generate host names, specify in the configuration file an `option host-name` statement with a value containing certain key values in addition to any characters that are valid for the `host-name` option (see the table below). The `option host-name` statement can be specified for example at the top level, in a `subnet` statement, or in a `host` statement.

The key values are as follows. You can include more than one in the same `host-name` value.

**Note:** Some of these do not by themselves generate a unique identifier.

Key	Meaning
%A	First byte of the host's IP address. Example: for address 10.24.25.201, the key would return 10.
%B	Second byte of the host's IP address. Example: for address 10.24.25.201, the key would return 24.
%C	Third byte of the host's IP address. Example: for address 10.24.25.201, the key would return 25.
%D	Fourth byte of the host's IP address. Example: for address 10.24.25.201, the key would return 201.
%H	Host part of the host's IP address. Example: for address 10.24.25.201 with subnet mask 255.255.0.0, the key would return 6601.
%I	Client Identifier sent by the host. (in hex). For example: 0174657374.
%-I	Client ID as above, except that hyphens (-) are used to separate each byte.
%M	MAC address of the host.
%-M	MAC address of the host, as above, except that hyphens (-) are used to separate each byte.
%N	Host name sent by the client, if any. If none, "Host".
%P	Printable characters from the client ID. For example: if the client ID was 0174657374, the 01 is thrown away and the resulting hostname is "test".
%S	Subnet part of the host's IP address. Example: for address 10.24.25.201 with subnet mask 255.255.0.0, the key would return 102400.



<code>%-S</code>	Subnet part of the host's IP address, as above, except that hyphens (-) are used to separate each byte. For example: 10-24-0-0.
------------------	---

You can intersperse string constants such as hyphens between key definitions. However, if the generated host name exceeds 63 characters, it is truncated. Here is an example host-name statement:

```
option host-name "Host%H-%-S";
```

For a lease pool defined with an address range of 192.168.11.6 through 192.168.11.10 and a subnet mask of 255.255.255.0, the DHCP server generates the following host names:

```
Host6-192-168-11-0
Host7-192-168-11-0
Host8-192-168-11-0
Host9-192-168-11-0
Host10-192-168-11-0
```

The `%N` key allows you to use the host name as sent by the client (option 12) and then add something unique to it to generate a unique name. For example, if multiple clients all send the name "dilbert" you can make them unique by appending the MAC (hardware) address, as follows:

```
deny name-by-client;
option host-name "%N-%M";
```

This would generate the host name "dilbert-010203040506" for a client with hardware address 01:02:03:04:05:06.

## Configuration File Declarations and Parameters

The below table describes the declarations and parameters you can use in a configuration file.

Statement	Description
add	Use the add statement to add a client to the class whose name is specified in <code>class-name</code> .

	<p><b>Important!</b> Because this statement executes after IP address allocation is completed, class membership caused by this statement cannot be used in the address allocation process.</p> <pre>add "class-name";</pre>
algorithm	<p>Used only inside of key declarations, the <code>algorithm</code> statement specifies the algorithm to use for <b>Error! Reference source not found.</b> on dynamic DNS updates. The only supported algorithm is "hmac-md5". This statement is optional.</p> <pre>algorithm "hmac-md5";</pre>
allow and deny	<p>Use the <code>allow</code> and <code>deny</code> statements to control the behavior of the DHCP server.</p> <p>The <code>allow</code> and <code>deny</code> keywords have different meanings depending on the context.</p> <ul style="list-style-type: none"> <li>• In a pool context, use these keywords to set up access lists for address allocation pools.</li> <li>• In other contexts, use these keywords to control general server behavior with respect to clients based on scope.</li> </ul>
allow and deny in scope	<p>These <code>allow</code> and <code>deny</code> statements work the same way whether the client is sending a DHCPDISCOVER or a DHCPREQUEST message,</p> <ul style="list-style-type: none"> <li>• an address is allocated to the client (either the old requested address or a new address), and then,</li> <li>• that address is tested to see if it is okay for the client to have it.</li> </ul>

If the client requested it, and it is not okay, the server sends a DHCPNAK message. Otherwise, the server does not respond to the client. If it is okay to give the address to the client, the server sends a DHCPACK message.

**Note:** These are not recommended for use inside `pool` declarations. See the *Pool Permit Lists* section for an important note.

Use the `unknown-clients` flag to tell the DHCP server to dynamically assign addresses to unknown clients or to not assign addresses to unknown clients. An unknown client is one that does not have a `host` declaration. The default is to `allow` dynamic address assignments to unknown clients.

```
allow unknown-clients;      deny unknown-clients;
```

Use the `bootp` flag to tell the DHCP server to respond to bootp queries or to not respond to bootp queries. The default is to *allow* bootp queries.

```
allow bootp;               deny bootp;
```

Use the `dynamic-bootp` flag to tell the DHCP server to dynamically assign addresses to bootp clients or to not do so. The default is to *allow* dynamic bootp for IP addresses declared in pool declarations. The default for `range` statements outside of pool declarations is set by the presence or absence of the `dynamic-bootp` keyword. `Deny dynamic-bootp` overrides the `dynamic-bootp range` key word.

```
allow dynamic-bootp;      deny dynamic-bootp;
```

Use the `booting` flag to tell the DHCP server to respond to queries from a particular client or to not respond to queries from a particular client. The default is to `allow booting`. If it is disabled for a particular client, that client will not be able to get an address from the DHCP server.

```
allow booting;      deny booting;
```

Use the `dynamic-update` flag to tell the DHCP server to perform dynamic DNS updates or to not perform them. The default is to `deny dynamic DNS updates`.

```
allow dynamic-update;  deny dynamic-update;
```

Use the `name-by-client` flag to tell the DHCP server to determine the hostname and Fully Qualified Domain Name (FQDN) for dynamic DNS updates from information sent by the client or from information in the configuration file. The default is to `deny use of client-specified information`.

```
allow name-by-client;  deny name-by-client;
```

Use the `dhcpinform` flag to tell the DHCP server to respond to DHCPINFORM messages or to not respond. The default is to `allow DHCPINFORM messages for authoritative subnets, and to deny DHCPINFORM messages for non-authoritative subnets`.

```
allow dhcpinform;      deny dhcpinform;
```

Use the `update-A-record` flag to tell the DHCP server to update the A resource record or not when performing DNS updates (the PTR resource record is always updated). The default is to `deny updating the A resource record`.

```
allow update-A-record;  deny update-A-record;
```

	<p>Use the <code>ras-servers</code> flag to tell the DHCP server to respond to queries from Microsoft Windows RAS Servers or to not respond to NT RAS queries. The default is to allow NT RAS queries.</p> <pre>allow ras-servers;      deny ras-servers;</pre> <p>Allow/deny <code>ras-servers</code> is supported for backward compatibility. The way to do <code>deny ras-servers</code> in modern versions of MultiNet is to use a conditional statement:</p> <pre>if substring (option dhcp-client-identifier, 1,3) = "RAS" {     deny booting; }</pre>
<p>allow and deny in pool declarations</p>	<p>See the <i>Pool Permit Lists</i> section for discussion, defaults, and important notes.</p> <p>Use <code>known clients</code> to allow or prevent allocation from this pool to any client that has a host declaration. A client is known if it has a host declaration in <b>any</b> scope.</p> <pre>allow known clients;    deny known clients;</pre> <p>Use <code>unknown clients</code> to allow or prevent allocation from this pool to any client that has no host declaration.</p> <pre>allow unknown clients;  deny unknown clients;</pre> <p>Use members of "class" to allow or prevent allocation from this pool to any client that is a member of the named class.</p>

	<pre>allow members of "class-name"; deny members of "class-name";</pre> <p>Use <code>dynamic bootp clients</code> to allow or prevent allocation from this pool to any BOOTP client.</p> <pre>allow dynamic bootp clients; deny dynamic bootp clients;</pre> <p>Use <code>all clients</code> to allow or prevent allocation from this pool to all clients. You can use this, for example, when you want to write a pool declaration but you want to hold it in reserve; or when you want to renumber your network quickly, and thus want the server to force all clients that have been allocated addresses from this pool to obtain new addresses immediately when they next renew their leases.</p> <pre>allow all clients;          deny all clients;</pre>
<pre>always- broadcast</pre>	<p>Use the <code>always-broadcast</code> statement to cause the DHCP server to always broadcast its responses. This feature is to handle clients who do not set the broadcast flag in their requests and yet require a broadcast response. We recommend you restrict the use of this feature to as few clients as possible.</p> <pre>always-broadcast flag;</pre>
<pre>always-reply- rfc1048</pre>	<p>Some BOOTP clients expect RFC 1048-style responses, but do not follow RFC 1048 rules when sending their requests. You can determine if a client is having this problem:</p> <ul style="list-style-type: none"> <li>• if it is not getting the options you have configured for it, and</li> <li>• if you see in the server log the message "(non-rfc1048)" printed with each BOOTREQUEST that is logged.</li> </ul>

	<p>If you want to send RFC 1048 options to this kind of client, set the <code>always-reply-rfc1048</code> option in that client's host declaration. The DHCP server responds with an RFC 1048-style vendor options field. This flag can be set in any scope, and affects all clients covered by that scope.</p> <pre>always-reply-rfc1048 flag;</pre>
<p>[not] authoritative</p>	<p>When the DHCP server receives a DHCPREQUEST message from a DHCP client requesting a specific IP address, the DHCP protocol requires that the server determine whether the IP address is valid for the network to which the client is attached. If the address is not valid, the DHCP server should respond with a DHCPNAK message, forcing the client to acquire a new IP address.</p> <p>To make this determination for IP addresses on a particular network segment, the DHCP server must have complete configuration information for that network segment. Unfortunately, it is not safe to assume that DHCP servers are configured with complete information. Therefore, the DHCP server normally assumes that it does not have complete information, and thus is not sufficiently authoritative to safely send DHCPNAK messages as required by the protocol.</p> <p>This default assumption should not be true for any network segment that is in the same administrative domain as the DHCP server. For such network segments, the <code>authoritative</code> statement should be specified, so that the server sends DHCPNAK messages as required by the protocol. If the DHCP server receives requests only from network segments in the same administrative domain, you can specify the <code>authoritative</code> statement at the top of the configuration file (in the global scope).</p> <pre>authoritative; not authoritative;</pre>
<p>class</p>	<p>This statement groups clients together based on information they send. A client can become a member of a class in the following ways:</p>

	<ul style="list-style-type: none"> <li>• through an add statement</li> <li>• based on the class's matching rules</li> <li>• because the client matches a subclass of that class</li> </ul> <p>Class-name is the name of the class and is used in:</p> <ul style="list-style-type: none"> <li>• add statements</li> <li>• members of permit statements</li> <li>• <i>subclass</i> declarations for subclasses of the named class</li> </ul> <p>When a packet is received from a client, every class declaration is examined for a match, match if, or spawn statement. That statement is checked to see if the client is a member of the class.</p> <p>The class declaration statements are lease limit, match, match if, and spawn with.</p> <pre>class "class-name" {[ statements ][ declarations ]}</pre>
default-lease-time	<p>Time is the length (in seconds) that the DHCP server assigns to a lease if the requesting client did not ask for a specific amount of time for the lease to be active. The infinite lease value is "infinite". The default is 43,200 seconds (12 hours).</p> <p>You should set the value of default-lease-time NO larger than the value of max-lease-time.</p> <pre>default-lease-time time;</pre>
dynamic-bootp-lease-cutoff	<p>Use the dynamic-bootp-lease-cutoff statement to set the ending time for all leases dynamically assigned to BOOTP clients. By default, the DHCP server assigns infinite leases to all BOOTP clients because they do not have any way of renewing leases, and do not know that their leases could expire.</p>



	<p>However, it may make sense to set a cutoff date for all BOOTP leases. For example, the end of a school term, or the time at night when a facility is closed and all machines are required to be powered off.</p> <p>Date should be the date all assigned BOOTP leases will end. The date is specified in the form:</p> <pre>W YYYY/MM/DD HH:MM:SS</pre> <p>W is the day of the week, from zero (Sunday) to six (Saturday). YYYY is the year, including the century. MM is the number of the month, from 01 to 12. DD is the day of the month, counting from 01. HH is the hour, from 00 to 23. MM is the minute, from 00 to 59. SS is the second, from 00 to 59.</p> <p>The time is always in Greenwich Mean Time, not local time.</p> <pre>dynamic-bootp-lease-cutoff date;</pre>
<code>dynamic-bootp-lease-length</code>	<p>Use the <code>dynamic-bootp-lease-length</code> statement to set the length of leases dynamically assigned to BOOTP clients. You may be able to assume that a lease is no longer in use if its holder has not used BOOTP or DHCP to get its address within a certain time period. The length of the time period is your judgment call.</p> <p>Specify length in seconds. The infinite lease value is “infinite”. If a BOOTP client reboots during a timeout period, the lease duration is reset to length so a BOOTP client that boots frequently never loses its lease. This parameter should be adjusted with extreme caution. The default is an infinite lease.</p>

	<pre>dynamic-bootp-lease-length <i>length</i>;</pre>
filename	<p>Use the <code>filename</code> statement to specify the name of the initial boot file that is to be loaded by a client. The <code>filename</code> should be recognizable to whatever file transfer protocol the client can be expected to use.</p> <pre>filename <u>filename</u>;</pre>
fixed-address	<p>To make a static IP address assignment for a client, the client must match a <code>host</code> declaration, as described later. In addition, the <code>host</code> declaration must contain a <code>fixed-address</code> declaration. A <code>fixed-address</code> declaration specifies one or more IP addresses or domain names that resolve to IP addresses. If a client matches a <code>host</code> declaration, and one of the IP addresses specified in the <code>host</code> declaration is valid for the network segment to which the client is connected, the client is assigned that IP address.</p> <p>A static IP address assignment overrides a dynamically assigned IP address that is valid on that network segment. That is, if a new static mapping for a client is added after the client has a dynamic mapping, the client cannot use the dynamic mapping the next time it tries to renew its lease. The DHCP server will not assign an IP address that is not correct for the network segment to which the client is attached and will not override a valid dynamic mapping for one network segment based on a static mapping that is valid on a different network segment.</p> <p>You can specify a domain name instead of an IP address in a <code>fixed-address</code> declaration. However, you should do this only for long-lived domain name records — the DHCP server only looks up the record on startup. So, if the record changes while the server is running, the server continues to use the record's former value.</p> <pre>fixed-address <i>address</i> [, ..., <i>address</i>];</pre>

<p>get-lease-hostnames</p>	<p>Use the <code>get-lease-hostnames</code> statement to tell the DHCP server to look up the domain name corresponding to each address in the lease pool and use that address for the DHCP hostname option.</p> <p>If <code>flag</code> is true, the lookup is done for all addresses in the current scope.</p> <p>If <code>flag</code> is false (the default), lookups are not done.</p> <pre>get-lease-hostnames <i>flag</i>;</pre>
<p>group</p>	<p>Use the <code>group</code> statement to apply one or more parameters to a group of declarations. You can use it to group hosts, shared networks, subnets, or other groups.</p> <pre>group {[statements] [declarations]}</pre>
<p>hardware</p>	<p>Use the <code>hardware</code> clause inside a <code>host</code> statement to specify the network hardware address of a BOOTP or DHCP client.</p> <p><code>hardware-type</code> must be the name of a physical hardware interface type. Ethernet, Token-Ring, and FDDI are the only recognized types.</p> <p>The <code>hardware-address</code> should be a set of hexadecimal octets (numbers from 0 through ff) separated by colons (:).</p> <pre>hardware hardware-type hardware-address;</pre>
<p>host</p>	<p>The <code>host</code> declaration provides information about a particular client.</p>

Name should be a unique name for the `host` declaration, but a specific meaning is not required. If the `use-host-decl-names` flag is enabled, name is sent in the `host-name` option if no `host-name` option is specified.

`Host` declarations match DHCP or BOOTP clients based on either the client's hardware address or the `dhcp-client-identifier` option that the client sends. BOOTP clients do not normally send a `dhcp-client-identifier` option. So, you must use the hardware address for all clients that might send BOOTP protocol requests.

The `host` declaration has three purposes:

- to assign a static IP address to a client
- to declare a client as "known"
- to specify a scope in which statements can be executed for a specific client

You can make the DHCP server treat some DHCP clients differently from others if `host` declarations exist for those clients. Any request coming from a client that matches a `host` declaration is considered to be from a "known" client. Requests that do not match any `host` declaration are considered to be from "unknown" clients. You can use this knowledge to control how addresses are allocated.

It is possible to write more than one `host` declaration for a client. If you want to assign more than one static address to a given client, you can either specify more than one address in the `fixed-address` statement or you can write multiple `host` declarations.

Multiple `host` declarations are needed if the client has different requirements (scopes) on different subnets. For each IP address that requires a different scope, one `host` declaration should exist. A client can be in the scope of only one `host` declaration at a time. `Host` declarations with static address assignments are in scope for a client only if one of the address assignments is

	<p>valid for the network segment to which the client is connected. If you want to boot a client using static addresses on some subnets, and using dynamically assigned addresses on other subnets, you need to write a <code>host</code> declaration with no <code>fixed-address</code> statement. There can be only one such <code>host</code> declaration per client. Its scope is used whenever that client receives a dynamically assigned address.</p> <pre>host name { [statements] [declarations] }</pre>
if	<p>The <code>if</code> statement conditionally executes statements based on the values the client sends or other information. See the <i>Conditional Behavior</i> section for more information.</p> <pre>if boolean-expression { [statements] } [elseif boolean-expression { [statements] }] [else { [statements] } ]</pre>
invalid-ddns-chars	<p>This statement specifies how DHCP should handle invalid characters in the hostname for Dynamic DNS updates (DDNS).</p> <p><code>fail</code> tells DHCP to display a message and not perform any DNS updates if there are any invalid characters in the hostname. This is the default.</p> <pre>invalid-ddns-chars fail;</pre> <p><code>discard</code> tells DHCP to throw away the invalid characters in the hostname.</p> <pre>invalid-ddns-chars discard;</pre> <p><code>replace</code> tells DHCP to replace the invalid characters with the specified character(s). If none are specified, the default replacement character is the hyphen ('-').</p>

	<pre>invalid-ddns-chars replace ["characters"];</pre>
key	<p>The key declaration specifies keys to use for Transaction Signatures (TSIG) to sign dynamic DNS updates. See the <i>Transaction Signatures</i> section.</p> <pre>key _ip-address_ {   [ algorithm "algorithm-name"; ]   key-id "key-name";   secret "key"; }</pre>
key-id	<p>Used only inside of key declarations, the key-id statement specifies the name of the key to use for Transaction Signatures (TSIG) on dynamic DNS updates. This key name must match the name that the DNS server is using (as specified in <code>named.conf</code>).</p> <pre>key-id "key-name";</pre>
lease limit	<p>This statement causes the DHCP server to limit the number of members of a class that can hold a lease at any one time. This limit applies to all addresses the DHCP server allocates in the class, not just addresses on a particular network segment.</p> <ul style="list-style-type: none"> <li>• If a client is a member of more than one class with lease limits, the server assigns the client an address based on either class.</li> <li>• If a client is a member of one or more classes with limits and one or more classes without limits, the classes without limits are not considered.</li> </ul> <pre>lease limit limit;</pre>
lease-scan-interval	<p>This statement specifies how frequently to scan for expired leases. The default is 60 seconds.</p> <pre>lease-scan-interval seconds;</pre>

match	<p><code>data-expression</code> is evaluated using the contents of a client's request. If it returns a value that matches a subclass of the class in which the <code>match</code> statement appears, the client is considered a member of both the subclass and the class.</p> <pre>match data-expression;</pre>
match if	<p><code>boolean-expression</code> is evaluated when the server receives a packet from the client. If it is true, the client is considered a member of the class. The <code>boolean-expression</code> may depend on the contents of the packet the client sends.</p> <pre>match if boolean-expression;</pre>
max-delayed-acks	<p>Use the <code>max-delayed-acks</code> statement to specify the maximum number of DHCPACKs to batch up. The default is 8. To disable the delaying of DHCPACKs, specify a value of 1.</p> <p>To improve performance under very heavy loads, the DHCP server delays sending DHCPACK messages by up to 2 seconds. All DHCPACKs accumulated in that time are sent in a batch.</p> <pre>max-delayed-acks count;</pre>
max-lease-time	<p>Use the <code>max-lease-time</code> statement to assign the maximum amount of time (in seconds) to a lease. The only exception to this is Dynamic BOOTP lease lengths because they are not specified by the client and are not limited by this maximum. The infinite lease value is "infinite". The default is 86,400 seconds (24 hours).</p>

	<p><b>Note:</b> You should set the value of <code>max-lease-time</code> at least as large as <code>default-lease-time</code>.</p> <pre>max-lease-time time;</pre>
<p><code>min-lease-time</code></p>	<p>Use the <code>min-lease-time</code> statement to assign the minimum length in seconds to a lease. The infinite lease value is “infinite”. By default, there is no minimum.</p> <p><code>min-lease-time</code> should be less than or equal to <code>default-lease-time</code> and <code>max-lease-time</code>.</p> <pre>min-lease-time time;</pre>
<p><code>min-secs</code></p>	<p>Use the <code>min-secs</code> statement to assign the minimum amount of time (in seconds) it takes for the DHCP server to respond to a client’s request for a new lease.</p> <p>The number of seconds is based on what the client reports in the <code>secs</code> field of the requests it sends. The maximum value is 255 seconds. Usually, setting this to one second results in the DHCP server not responding to the client’s first request, but always responding to the client’s second request.</p> <p>You can use the <code>min-secs</code> statement to set up a secondary DHCP server to never offer an address to a client until the primary server has been given a chance to do so. If the primary server is down, the client binds to the secondary server; otherwise, clients should always bind to the primary.</p>



	<p><b>Note:</b> This does not permit a primary server and a secondary server to share a pool of dynamically-allocatable addresses.</p> <p>See information about Safe-failover in this chapter.</p> <pre>min-secs seconds;</pre>
next-server	<p>Use the <code>next-server</code> statement to specify the host address of the server from where the client will load the initial boot file (specified in the <code>filename</code> statement).</p> <p><code>name</code> should be a numeric IP address or a domain name. The DHCP server's IP address is used if no <code>next-server</code> parameter applies to a given client.</p> <pre>next-server name;</pre>
one-lease-per-client	<p>Use the <code>one-lease-per-client</code> statement to have the server free any other leases the client holds when the client sends a DHCPREQUEST for a particular lease.</p> <p>This presumes the client has forgotten any lease not mentioned in the DHCPREQUEST. For example, the client has only a single network interface and it does not remember leases it is holding on networks to which it is not currently attached. Neither of these assumptions are guaranteed or provable, so use caution in the use of this statement.</p> <pre>one-lease-per-client flag;</pre>

option	<p>This statement specifies actual DHCP protocol options to send to the client. The option statement is described in the <i>DHCP Options</i> section.</p>
option definition	<p>This statement assigns a name and a type to an option code. See the <i>Defining New Options</i> section for more information.</p> <pre data-bbox="441 512 1476 548">option name code code = definition;</pre>
option space	<p>This statement specifies a new option space. This declaration must precede all definitions for options in the space being specified. <code>space-name</code> should be the name of the option space. Currently three option space names are predefined:</p> <pre data-bbox="441 869 688 961">dhcp (default) agent server</pre> <p>If an option name is specified without an option space, it is assumed the name refers to an option in the <code>dhcp</code> option space. For example, the option names <code>dhcp.routers</code> and <code>routers</code> are equivalent.</p> <pre data-bbox="441 1262 862 1297">option space space-name;</pre>
ping	<p>The DHCP server uses ping to check if a particular IP address is in use by sending a packet of information and waiting for a response. This statement turns ping on and off. The default is ON.</p> <pre data-bbox="441 1570 618 1606">ping flag;</pre>
ping-retries	<p>This statement defines the number of times the DHCP server pings an IP address before it concludes that the address is not in use. The default is 1.</p> <pre data-bbox="441 1835 776 1871">ping-retries count;</pre>

ping-timeout	<p>This statement defines the time (in seconds) that ping should wait for a response. The default is 1 second.</p> <pre>ping-timeout <i>time</i>;</pre>
pool	<p>This statement specifies an address pool from which IP addresses can be allocated. This pool can be customized to have its own permit list to control client access and its own scope to declare pool-specific parameters. You can put <code>pool</code> declarations within <code>subnet</code> declarations or within <code>shared-network</code> declarations. You can use the <code>range</code> declaration to specify the addresses in a particular pool.</p> <ul style="list-style-type: none"> <li>• For <code>subnet</code> declarations: specified addresses must be correct within the <code>pool</code> declaration within which it is made.</li> <li>• For <code>shared-network</code> declarations: specified addresses must be on subnets that were previously specified within the same <code>shared-network</code> declaration.</li> </ul> <pre>pool {[permit <i>list</i>][range <i>declaration</i>][statements]}</pre>
range	<p>For any subnet on which addresses are assigned dynamically, there must be at least one <code>range</code> declaration. The <code>range</code> declaration specifies that the server may allocate to DHCP clients every address, from <i>low-address</i> to <i>high-address</i>. You can specify a single IP address by omitting <i>high-address</i>.</p> <p>All IP addresses in the range should be on the same subnet. If the <code>range</code> declaration appears within a <code>subnet</code> declaration, all addresses should be on the declared subnet. If the <code>range</code> declaration appears within a <code>shared-network</code> declaration, all addresses should be on subnets already declared within the <code>shared-network</code> declaration.</p> <p>You may specify the <code>dynamic-bootp</code> flag if addresses in the specified range can be dynamically assigned to both BOOTP and DHCP clients.</p>

	<p><b>Note:</b> The <code>dynamic-bootp</code> flag was deprecated in MultiNet 5.1's version of the DHCP server in favor of declaring the address within a pool and specifying in the permit list that dynamic allocation for BOOTP clients is permitted.</p> <pre>range [dynamic-bootp] low-address [high-address];</pre>
<p>requested-options-only</p>	<p>Use the <code>requested-options-only</code> statement to send just the options requested by the client. To send a specific set of options, set <code>requested-options-only</code> to true and specify the <code>dhcp-parameter-request-list</code> option.</p> <p>The following sends only the <code>subnet-mask</code>, <code>routers</code>, and <code>domain-name-servers</code> options to the client (assuming they are defined in the configuration file):</p> <pre>host restricted {     hardware ethernet 01:02:03:04:05:06;     option dhcp-parameter-request-list 1, 3, 6;     requested-options-only true; }</pre> <p>We recommend you restrict the use of this feature to as few clients as possible.</p> <pre>requested-options-only flag;</pre>
<p>secret</p>	<p>Used only inside of key declarations, the <code>secret</code> statement specifies the actual secret key to use for <b>Error! Reference source not found.</b> on dynamic</p>

	<p>DNS updates. The format is base-64. The value must match the key used by the DNS server (as specified in <code>named.conf</code>).</p> <pre>secret "key";</pre>
secure-ddns	<p>Use the <code>secure-ddns</code> statement to cause the DHCP server to use <b><i>Error! Reference source not found.</i></b> to sign dynamic DNS updates. The default is to not use TSIG.</p> <pre>secure-ddns flag;</pre>
server-identifier	<p>The <code>server-identifier</code> statement is equivalent to the <code>dhcp-server-identifier</code> option. See the <code>dhcp-server-identifier</code> option for more information</p> <pre>server-identifier hostname;</pre>
server-name	<p>Use the <code>server-name</code> statement to inform the client of the server's name from which it is booting. <code>name</code> should be the name provided to the client.</p> <pre>server-name name;</pre>
shared-network	<p>Use this statement to inform the DHCP server that some IP subnets share the same physical network. Declare all subnets in the same shared network within a <code>shared-network</code> statement.</p> <p><i>Parameters</i> specified in the <code>shared-network</code> statement will be used when booting clients on those subnets unless parameters provided at the subnet or host level override them. If more than one subnet in a shared network has addresses available for dynamic allocation, those addresses are collected into a common pool. There is no way to specify which subnet of a shared network a client should boot on.</p>

	<p>Name should be the name of the shared network. Make the name descriptive as it will be used when printing debugging messages. Give it a syntax of a valid domain name (although it will never be used as such), or any arbitrary name enclosed in quotation marks.</p> <pre>shared-network name {[statements] [declarations]}</pre>
site-option-space	<p>Use the <code>site-option-space</code> statement to determine the option space from which site-local options are taken. Site-local options have codes ranging from 128 to 254. If no <code>site-option-space</code> is specified, site-specific options are taken from the default option space.</p> <pre>site-option-space option-space;</pre>
spawn with	<p><code>data-expression</code> must evaluate to a non-null value for the server to look for a subclass of the class that matches the evaluation.</p> <ul style="list-style-type: none"> <li>• If such a subclass exists, the client is considered a member of both the subclass and the class.</li> <li>• If no such subclass exists, one is created and recorded in the lease database, and the client is considered a member of the new subclass as well as the class.</li> </ul> <pre>spawn with data-expression;</pre>
subclass	<p>This statement specifies a subclass of the class named by <code>class-name</code>. <code>Class-data</code> should be either</p> <ul style="list-style-type: none"> <li>• a text string enclosed in quotes, or</li> <li>• a list of bytes expressed in hexadecimal, separated by colons.</li> </ul> <p>Clients match subclasses after evaluating the <code>match</code> or <code>spawn with</code> statements in the <code>class</code> declaration for <code>class-name</code>. If the evaluation matches <code>class-data</code>, the client is a member of the subclass and the class.</p>

```
subclass "class-name" class-data;
subclass "class-name" class-data {
    [statements]
}
```

subnet

This statement contains information specific to a subnet. The information communicates the following to DHCP:

- Enough information for DHCP to determine if an IP address is on that subnet.
- What the subnet-specific parameters are.
- What addresses may be dynamically allocated to clients booting on that subnet.

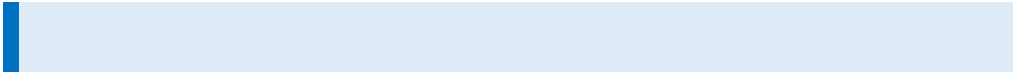
Use the `range` declaration to specify what addresses are available to be dynamically allocated to clients booting on the subnet.

Two things are required to define a subnet:

- The `subnet-number`
- The `netmask`

The `subnet-number` and the `netmask` entry is an IP address or domain name that resolves to the `subnet-number` or the `netmask` of the subnet being described. The `subnet-number` and the `netmask` are enough to determine if any given IP address is on the specified subnet.

**Note:** A `netmask` must be given with every `subnet` declaration. If there is any variance in subnet masks at a site, use a `subnet-mask` option statement in each `subnet` declaration to set the desired subnet mask. The `subnet-mask` option statement overrides the subnet mask declared in the `subnet` statement.

	 <pre data-bbox="446 409 1469 472">subnet subnet-number netmask <i>netmask</i> { [<i>statements</i>] [<i>declarations</i>] }</pre>
<pre data-bbox="151 531 397 588">use-host-decl- names</pre>	<p data-bbox="446 531 1453 651">If the <code>use-host-decl-names</code> parameter is true, the name provided for each host declaration is given to the client as its hostname. The default is false. For example:</p> <pre data-bbox="446 745 1112 976">group {   use-host-decl-names on;   host joe {     hardware ethernet 08:00:2b:4c:29:32;     fixed-address joe.example.com;   } }</pre> <p data-bbox="446 1050 641 1081">is equivalent to</p> <pre data-bbox="446 1186 1096 1344">host joe {   hardware ethernet 08:00:2b:4c:29:32;   fixed-address joe.example.com;   option host-name "joe"; }</pre> <p data-bbox="446 1417 1429 1501">An option <code>host-name</code> statement within a host declaration overrides the use of the name in the host declaration.</p> <pre data-bbox="446 1596 885 1627">use-host-decl-names <i>flag</i>;</pre>
<pre data-bbox="151 1690 414 1785">use-lease-addr- for-default- route</pre>	<p data-bbox="446 1690 1445 1816">If the <code>use-lease-addr-for-default-route</code> parameter is true in a given scope, the IP address of the lease being assigned is sent to the client instead of the value specified in the routers option (or sending no value at all).</p>



	<p>This causes some clients to ARP for all IP addresses, which can be helpful if your router is configured for proxy ARP.</p> <p>If <code>use-lease-addr-for-default-route</code> is enabled and an option routers statement are both in scope, <code>use-lease-addr-for-default-route</code> is preferred.</p> <pre>use-lease-addr-for-default-route flag;</pre>
<code>user-class</code>	<p>This statement has been deprecated in favor of the more powerful <code>class</code> statement. See the example in the <i>Vendor Encapsulated Options</i> section.</p>
<code>vendor-class</code>	<p>This statement has been deprecated in favor of the more powerful <code>class</code> statement. See the example in the <i>Vendor Encapsulated Options</i> section.</p>
<code>vendor-option-space</code>	<p>Use the <code>vendor-option-space</code> statement to instruct the server to construct a <code>vendor-encapsulated-options</code> option using all the defined options in the option space. If no <code>vendor-encapsulated-options</code> option is defined, the server sends this option to the client, if appropriate.</p> <pre>vendor-option-space option-space;</pre>

## Expressions

The DHCP server can evaluate expressions while executing statements. The DHCP server's expression evaluator returns the following types:

- A *boolean*, a true or false (on or off) value.
- An *integer*, a 32-bit quantity that may be treated as signed or unsigned, depending on the context.
- A *string of data*, a collection of zero or more bytes. Any byte value is valid in a data string — the DHCP server maintains a length rather than depending on a NUL termination.

Expression evaluation is performed when a request is received from a DHCP client. Values in the packet sent by the client can be extracted and used to determine what to send back to the client. If the expression refers to a field or option in the packet for which there is no value, the result is null. Null values are treated specially in expression evaluation. A Boolean expression that returns a null value is considered false. A data expression that returns a null value generally results in the statement using the value not having any effect.

The following is an example of using four types of expressions to produce the name of a PTR record for the IP address being assigned to a client:

```
concat (binary-to-ascii (10, 8, ".", reverse (1, leased-address)),
        ".in-addr.arpa.");
```

## Boolean Expressions

The following are the boolean expressions supported by DHCP.

boolean-expression-1 and boolean-expression-2	The <code>and</code> operator evaluates to true if both boolean expressions evaluate to true. The <code>and</code> operator evaluates to false if either boolean expression does not evaluate to true. If either of the boolean expressions is null, the result is null.
boolean-expression-1 or boolean-expression-2	The <code>or</code> operator evaluates to true if either of the boolean expressions evaluate to true. The <code>or</code> operator evaluates to false if both of the boolean expressions evaluate to false. If either of the boolean expressions is null, the result is null.
check "class-name"	The <code>check</code> operator evaluates to true if the packet being considered comes from a client in the specified class. <code>Class-name</code> must be a string that corresponds to the name of a defined class.
data-expression-1 = data-expression-2	The <code>=</code> operator compares the results of evaluating two data expressions, evaluating to true if they are the same; evaluating to false if they are not. If one of the expressions is null, the result is null.
exists option-name	The <code>exists</code> expression evaluates to true if the specified option exists in the incoming DHCP packet.

known	The <code>known</code> expression evaluates to true if the client whose request is being processed is known; that is, if the client has a host declaration.
not boolean-expression	The <code>not</code> operator evaluates to true if the boolean expression evaluates to false. The <code>not</code> operator evaluates to false if the boolean expression evaluates to true. If the boolean expression evaluates to null, the result is null.

## Data Expressions

The following are the expressions supported by DHCP that return a data string.

<pre>binary-to-ascii (numeric-expr1,  numeric-expr2,  data-expr1, data-expr2)</pre>	<p><i>numeric-expr1</i>, <i>numeric-expr2</i>, <i>data-expr1</i>, and <i>data-expr2</i> are all evaluated as expressions and the results of those evaluations are used as follows.</p> <p>The <code>binary-to-ascii</code> operator converts the binary data in <i>data-expr2</i> into an ASCII string, using <i>data-expr1</i> as a separator. How the conversion is done is controlled by <i>numeric-expr1</i> and <i>numeric-expr2</i>.</p> <ul style="list-style-type: none"> <li>• <i>numeric-expr1</i> specifies the base to convert into. Any value 2 through 16 is supported. For example, a value of 10 would produce decimal numbers in the result.</li> <li>• <i>numeric-expr2</i> specifies the number of bits in <i>data-expr2</i> to treat as a single unit. The value can be 8, 16, or 32.</li> </ul> <p>This example converts the binary value of an IP address into its dotted decimal equivalent:</p> <pre>binary-to-ascii(10, 8, ".", 168364039)</pre> <p>The result would be the string "10.9.8.7".</p>
---	---

colon-separated hexadecimal list	A list of hexadecimal octet values, separated by colons, may be specified as a data expression. A single hexadecimal number, appearing in a context where a data string is expected, is interpreted as a data string containing a single byte.
<code>concat (data-expr1, data-expr2)</code>	<p><code>data-expr1</code> and <code>data-expr2</code> are evaluated and the concatenated result of these two evaluations is returned.</p> <ul style="list-style-type: none"> <li>• If either subexpression evaluates to null, the result is the value of the expression that did <b>not</b> evaluate to null.</li> <li>• If both expressions evaluate to null, the result is null.</li> </ul>
<code>encode-int (numeric-expr, width)</code>	<code>numeric-expr</code> is evaluated and encoded as a data string of the specified <code>width</code> , in network byte order (with the most significant byte first). If <code>numeric-expr</code> evaluates to null, the result is null.
<code>hardware</code>	<p>The <code>hardware</code> operator returns a data string whose first element is the <code>h<sub>type</sub></code> field of the packet being considered, and whose subsequent elements are the first <code>h<sub>len</sub></code> bytes of the <code>ch<sub>addr</sub></code> field of the packet.</p> <p>If there is no packet, or if the RFC 2131 <code>h<sub>len</sub></code> field is invalid, the result is null.</p> <p>Supported hardware types are:</p> <ul style="list-style-type: none"> <li>• ethernet (1)</li> <li>• token-ring (6)</li> <li>• fddi (8)</li> </ul>
<code>leased-address</code>	In any context where the processing client request has been assigned an IP address, this data expression returns that IP address.
<code>option option-name</code>	The <code>option</code> operator returns the contents of the specified option in the incoming DHCP packet.
<code>packet (offset, length)</code>	The <code>packet</code> operator returns the specified portion of the packet being considered. The <code>packet</code> operator returns a value of null

	<p>where no packet is being considered. <i>Offset</i> and <i>length</i> are applied to the contents of the packet as in the <code>substring</code> operator. The link-layer, IP, and UDP headers are not available.</p>
<p><code>reverse (numeric-expr1, data-expr2)</code></p>	<p><i>numeric-expr1</i> and <i>data-expr2</i> are evaluated. The result of <i>data-expr2</i> is reversed in place, using chunks of the size specified in <i>numeric-expr1</i>.</p> <p>For example, if <i>numeric-expr1</i> evaluates to four and <i>data-expr2</i> evaluates to twelve bytes of data, the <code>reverse</code> expression evaluates to twelve bytes of data constructed in the following way:</p> <ul style="list-style-type: none"> <li>• the last four bytes of the input data,</li> <li>• followed by the middle four bytes,</li> <li>• followed by the first four bytes.</li> </ul>
<p><code>substring (data-expr, offset, length)</code></p>	<p>The <code>substring</code> operator evaluates the data expression and returns the substring of the result of that evaluation that starts <i>offset</i> bytes from the beginning and continues for <i>length</i> bytes. <i>Offset</i> and <i>length</i> are numeric expressions.</p> <ul style="list-style-type: none"> <li>• If <i>data-expr</i>, <i>offset</i>, or <i>length</i> evaluate to null, the result is null.</li> <li>• If <i>offset</i> is greater than or equal to the length of the evaluated data, a zero-length data string is returned.</li> <li>• If <i>length</i> is greater than the remaining length of the evaluated data after <i>offset</i>, a data string containing all data from <i>offset</i> to the end of the evaluated data is returned.</li> </ul>
<p><code>suffix (data-expr, length)</code></p>	<p>The <code>suffix</code> operator evaluates <i>data-expr</i> and returns the last <i>length</i> bytes of that evaluation. <i>Length</i> is a numeric expression.</p> <ul style="list-style-type: none"> <li>• If <i>data-expr</i> or <i>length</i> evaluate to null, the result is null.</li> <li>• If <i>length</i> evaluates to a number greater than the length of the evaluated data, the evaluated data is returned.</li> </ul>

<code>"text"</code>	A text string, enclosed in quotes, may be specified as a data expression. The string returns the text between the quotes, encoded in ASCII.
---------------------	---

## Numeric Expressions

Numeric expressions evaluate to an integer. In general, the precision of numeric expressions is at least 32 bits. However, the precision of such integers may be more than 32 bits.

<code>extract-int (data-expr, width)</code>	The <code>extract-int</code> operator extracts an integer value in network byte order after evaluating <code>data-expr</code> . Width is the width in bits (either 8, 16, 32) of the integer to extract. If the evaluation of <code>data-expr</code> does not provide an integer of the specified size, a value of null is returned.
<code>number</code>	<code>Number</code> can be any numeric value between zero and the maximum representable size.

## DHCP Options

The Dynamic Host Configuration protocol allows the client to receive options from the DHCP server describing the network configuration and various services that are available on the network. When configuring the DHCP server, options must often be declared. The syntax for declaring options, and the names and formats of the options in the default `dhcp` option space that can be declared, are in the table below.

DHCP option statements always start with the keyword `option`, followed by an option name, followed by option data. Only options needed by clients must be specified.

An option name is an optional option space name followed by a period (".") followed by the option name. The default option space is `dhcp`. There are two other predefined option spaces: `agent` and `server`. You can also define option spaces of your own. See the sections *Relay Agent Information Option* and *Defining New Options* in this chapter.

Option data comes in these formats:

- The `ip-address` data type can be entered either as an explicit IP address (e.g., 239.254.197.10) or as a domain name (e.g., haagen.isc.org). When entering a domain name, be sure that the domain name resolves to the single IP address.
- The `int32` and `uint32` data types specify signed and unsigned 32-bit integers. The `int16` and `uint16` data types specify signed and unsigned 16-bit integers. The `int8` and `uint8` data types specify signed and unsigned 8-bit integers. Unsigned 8-bit integers are also sometimes referred to as octets.
- The `string` data type specifies an NVT ASCII string. It must be enclosed in quotation marks. For example, `option domain-name "isc.org";`
- The `flag` data type specifies a boolean value. Booleans can be either true (ON) or false (OFF). You can use `TRUE` and `FALSE`, or `ON` and `OFF`.
- The `data-string` data type specifies either an NVT ASCII string enclosed in quotation marks, or a series of octets specified in hexadecimal, separated by colons. For example, `option dhcp-client-identifier "CLIENT-FOO";` or `option dhcp-client-identifier 43:4c:49:54:2d:46:4f:4f;`

Strings and data-strings when enclosed in quotation marks can contain normal C-type characters such as “\t” for a tab.

If the option value is a list (such as for the `routes` option), you must list them in the configuration file in the order you want the client to use the values. The DHCP server does not re-order them.

Also, option data may be specified using an expression that returns a data string (see the *Expressions* section). The syntax is

```
option option-name = data-expression;
```

## Standard DHCP Options

This table describes the standard DHCP options. Underlined items indicate user input items. All of these options can be specified with the `dhcp` option space listed explicitly. For example:

```
option dhcp.bootfile-name "bootfile.lis";
```

Option	Description
--------	-------------

<pre>option all-subnets-local <i>flag</i>;</pre>	<p>Use this option to indicate whether or not to assume all subnets of the client's IP network use the same MTU as the client's subnet.</p> <p>ON means assume all subnets share the same MTU. OFF means assume some subnets have smaller MTUs.</p>
<pre>option arp-cache-timeout <i>uint32</i>;</pre>	<p>Use this option to identify the timeout (in seconds) for ARP cache entries.</p>
<pre>option bootfile-name <i>string</i>;</pre>	<p>Use this option to identify a bootstrap file. If this option is supported by the client, it should have the same effect as the <code>filename</code> declaration. BOOTP clients are unlikely to support this option. Some DHCP clients support it; others require it.</p>
<pre>option boot-size <i>uint16</i>;</pre>	<p>Use this option to specify the length in 512-octet blocks of the client's default boot image.</p>
<pre>option broadcast-address <i>ip-address</i>;</pre>	<p>Use this option to identify the broadcast address in use on the client's subnet. See STD 3 (RFC 1122), section 3.2.1.3 for legal values for broadcast addresses.</p>
<pre>option cookie-servers <i>ip-address</i> [, <i>ip-address</i> ...];</pre>	<p>Use this option to list RFC 865 cookie servers in order of preference.</p>
<pre>option default-ip-ttl <i>uint8</i>;</pre>	<p>Use this option to identify the default time-to-live the client should use on outgoing datagrams.</p>
<pre>option default-tcp-ttl <i>uint8</i>;</pre>	<p>Use this option to identify the default TTL to use when sending TCP segments. The minimum value is 1.</p>
<pre>option dhcp-client-identifier <i>data-string</i>;</pre>	<p>Use this option to specify a DHCP client identifier only in a <code>host</code> declaration. The DHCP server uses it to locate the <code>host</code> record by matching against the client identifier.</p>



<pre>option dhcp-max-message-size uint16;</pre>	<p>Use this option to specify the maximum length DHCP message that the client is able to accept. Use this option in the DHCP configuration file to supply a value when the client does not.</p> <p><b>Note:</b> Use this option with caution. Make sure that the client can accept a message of the specified size.</p>
<pre>option dhcp-parameter-request- list uint8[,uint8...];</pre>	<p>Use this option to request that the server return certain options. Use this option in the DHCP configuration file to override the client's list, or to supply a list when the client does not. The value is a list of valid DHCP option codes as listed in RFC 2132.</p>
<pre>option dhcp-server-identifier ip-address;</pre>	<p>Use this option to identify the value sent in the DHCP Server Identifier option. The value must be an IP address for the DHCP server, and must be reachable by all clients it is sent to.</p> <p>It is recommended to NOT use the <code>dhcp-server-identifier</code> option. The only reason to use it is to force a value other than the default value to be sent on occasions where the default value would be incorrect. The default value is the first IP address associated with the physical network interface on which the request arrived. The usual case where the <code>dhcp-server-identifier</code> option needs to be sent is when a physical interface has more than one IP address, and the one being sent by default is not appropriate for some or all clients served by that interface.</p>

	<p>Another case is when an alias is defined for the purpose of having a consistent IP address for the DHCP server, and it is desired that the clients use this IP address when contacting the server.</p>
<pre>option domain-name-servers ip-address [, ip-address ...];</pre>	<p>Use this option to list Domain Name System (STD 12, RFC 1035) name servers in order of preference.</p>
<pre>option domain-name string;</pre>	<p>Use this option to identify the domain name the client should use when resolving hostnames via the Domain Name System.</p>
<pre>option extensions-path string;</pre>	<p>Use this option to indicate the path-name of a file the client should load containing more options.</p>
<pre>option finger-server ip-address [, ip-address ...];</pre>	<p>Use this option to list the Finger servers in order of preference.</p>
<pre>option font-servers ip-address [, ip-address ...];</pre>	<p>Use this option to list X Window System Font servers in order of preference.</p>
<pre>option host-name string;</pre>	<p>Use this option to name the client. The name may or may not be qualified with the local domain name. It is preferable to use the domain-name option to specify the domain name. See RFC 1035 for character set restrictions.</p> <p>The host-name option is also used to specify a template for hostname generation. See the <i>Host Name Generation</i> section.</p>
<pre>option ieee802-3-encapsulation flag;</pre>	<p>If the interface is an Ethernet, use this option to indicate whether the client uses Ethernet Version 2 (RFC 894) or IEEE 802.3 (RFC 1042) encapsulation.</p>

	<p>OFF means use RFC 894 encapsulation.  ON means use RFC 1042 encapsulation.</p>
<pre>option ien116-name-servers ip-address [, ip-address ...];</pre>	<p>Use this option to list IEN 116 name servers in order of preference.</p>
<pre>option impress-servers ip-address [, ip-address ...];</pre>	<p>Use this option to list Imagen Impress servers in order of preference.</p>
<pre>option interface-mtu uint16;</pre>	<p>Use this option to identify what MTU value to use on this interface. The minimum legal value is 68.</p>
<pre>option ip-forwarding flag;</pre>	<p>Use this option to indicate if the client should configure its IP layer for packet forwarding.</p> <p>ON means disable forwarding.  OFF means enable forwarding.</p>
<pre>option irc-server ip-address [, ip-address ...];</pre>	<p>Use this option to list the IRC servers in order of preference.</p>
<pre>option log-servers ip-address [, ip-address ...];</pre>	<p>Use this option to list MIT-LCS UDP log servers in order of preference.</p>
<pre>option lpr-servers ip-address [, ip-address ...];</pre>	<p>Use this option to list RFC 1179 line printer servers in order of preference.</p>
<pre>option mask-supplier flag;</pre>	<p>Use this option to indicate whether or not the client should respond to subnet mask requests using ICMP.</p> <p>ON means do not respond to subnet mask requests.  OFF means respond to subnet mask requests.</p>
<pre>option max-dgram-reassembly uint16;</pre>	<p>Use this option to indicate the maximum size datagram the client should be prepared to reassemble. The minimum legal value is 576.</p>

<pre>option merit-dump <i>string</i>;</pre>	<p>Use this option to indicate the path-name of a file to which the client's core image should be dumped in the event of a client crash. The path is formatted as a character string using the NVT ASCII character set.</p>
<pre>option mobile-ip-home-agent <i>ip-address</i> [, <i>ip-address</i> ...];</pre>	<p>Use this option to list mobile IP home agents in order of preference. Usually there will be only one agent.</p>
<pre>option nds-context <i>data-string</i>;</pre>	<p>Use this option to identify the initial NDS context the client should use.</p>
<pre>option nds-servers <i>ip-address</i> [, <i>ip-address</i>...];</pre>	<p>Use this option to list Novell Directory Services servers in order of preference.</p>
<pre>option nds-tree-name <i>data-string</i>;</pre>	<p>Use this option to name the NDS tree the client will be contacting.</p>
<pre>option netbios-dd-server <i>ip-address</i> [, <i>ip-address</i> ...];</pre>	<p>Use this option to list RFC 1001/1002 NetBIOS Datagram Distribution servers in order of preference.</p>
<pre>option netbios-name-servers <i>ip-address</i> [, <i>ip-address</i> ...];</pre>	<p>Use this option to list RFC 1001/1002 NetBIOS Name Server name servers in order of preference.</p> <div data-bbox="760 1249 1469 1459" style="background-color: #e6f2ff; padding: 10px; border-left: 3px solid #0070c0;"> <p><b>Note:</b> NetBIOS is the same as WINS.</p> </div>
<pre>option netbios-node-type <i>uint8</i>;</pre>	<p>Use this option to configure configurable NetBIOS over TCP/IP clients as described in RFC 1001/1002. The value is a single octet identifying the client type.</p> <ul style="list-style-type: none"> <li>1 B-node: Broadcast - No WINS</li> <li>2 P-node: Peer - WINS only</li> <li>4 M-node: Mixed - Broadcast, then WINS</li> <li>8 H-node: Hybrid - WINS, then Broadcast</li> </ul>

<pre>option netbios-scope <i>string</i>;</pre>	<p>Use this option to specify the NetBIOS over TCP/IP scope parameter for the client as specified in RFC 1001/1002. See RFC1001, RFC1002, and RFC1035 for character-set restrictions.</p>
<pre>option nis-domain <i>string</i>;</pre>	<p>Use this option to specify the client's NIS (Sun Network Information Services) domain. Use the NVT ASCII character set to define the domain character string.</p>
<pre>option nis-servers ip-address [, ip-address ...];</pre>	<p>Use this option to list NIS servers in order of preference.</p>
<pre>option nisplus-domain <i>string</i>;</pre>	<p>Use this option to specify the client's NIS+ domain. Use the NVT ASCII character set to define the domain character string.</p>
<pre>option nisplus-servers ip-address [, ip-address ...];</pre>	<p>Use this option to list NIS+ servers in order of preference.</p>
<pre>option non-local-source-routing flag;</pre>	<p>Use this option to indicate if the client should configure its IP layer to allow forwarding of datagrams with non-local source routes.</p> <p>ON means disable forwarding. OFF means enable forwarding.</p>
<pre>option nntp-server ip-address [, ip-address ...];</pre>	<p>Use this option to list NNTP servers in order of preference.</p>
<pre>option ntp-servers ip-address [, ip-address ...];</pre>	<p>Use this option to list NTP (RFC 1305) servers in order of preference.</p>
<pre>option option-<i>nnn</i> <i>data-string</i>;</pre>	<p>Use this option to identify any DHCP option not listed here. <i>nnn</i> is the number of the option.</p>

<pre>option path-mtu-aging-timeout uint32;</pre>	<p>Use this option to specify the timeout to use (in seconds) when aging Path MTU values that were discovered by the mechanism defined in RFC 1191.</p>
<pre>option path-mtu-plateau-table uint16 [, uint16 ...];</pre>	<p>Use this option to specify a table of MTU sizes to use when performing Path MTU Discovery as defined in RFC 1191. The table is a list of 16-bit unsigned integers. You must list them in order from smallest to largest. The minimum MTU value cannot be smaller than 68.</p>
<pre>option perform-mask-discovery flag;</pre>	<p>Use this option to indicate whether or not the client should perform subnet mask discovery using ICMP.</p> <p>ON means do not perform mask discovery. OFF means perform mask discovery.</p>
<pre>option policy-filter ip-address ip-address [, ip-address, ip- address ...];</pre>	<p>Use this option to indicate the policy filters for non-local source routing. The filters consist of IP addresses and masks that indicate which destination/mask pairs to use when filtering incoming source routes.</p> <p>The client should discard any source routed datagram whose next-hop address does not match one of the filters. See STD 3 (RFC 1122) for more information.</p>
<pre>option pop-server ip-address [, ip-address ...];</pre>	<p>Use this option to list POP3 servers in order of preference.</p>
<pre>option resource-location-servers ip-address [, ip-address ...];</pre>	<p>Use this option to list RFC 887 Resource Location servers in order of preference.</p>
<pre>option root-path string;</pre>	<p>Use this option to specify the path-name that contains the client's root disk. The path is formatted as a character string using the NVT ASCII character set.</p>

<pre>option router-discovery <i>flag</i>;</pre>	<p>Use this option to indicate whether or not the client should solicit routers using the Router Discovery mechanism defined in RFC 1256.</p> <p>ON means do not perform router discovery. OFF means perform router discovery.</p>
<pre>option routers <i>ip-address</i> [, <i>ip-address</i> ...];</pre>	<p>Use this option to list IP addresses for routers on the client's subnet, listing the routers in order of preference.</p>
<pre>option router-solicitation- address <i>ip-address</i>;</pre>	<p>Use this option to identify the address where the client transmits router solicitation requests.</p>
<pre>option smtp-server <i>ip-address</i> [, <i>ip-address</i> ...];</pre>	<p>Use this option to list SMTP servers in order of preference.</p>
<pre>option static-routes <i>ip-address</i> <i>ip-address</i> [, <i>ip-address</i> <i>ip-</i> <i>address</i> ...];</pre>	<p>Use this option to specify a list of static routes that the client should install in its routing cache. If there are multiple routes to the same destination, you should list them in descending order of priority.</p> <p>The routes are made up of IP address pairs. The first address is the destination address; the second address is the router for the destination.</p> <p>The default route (0.0.0.0) is an illegal destination for a static route. Use the <code>routers</code> option to specify the default route.</p>
<pre>option streetwork-directory- assistance-server <i>ip-address</i> [, <i>ip-address</i> ...];</pre>	<p>Use this option to list the StreetTalk Directory Assistance (STDA) servers in order of preference.</p>
<pre>option streetwork-server <i>ip-address</i> [, <i>ip-address</i> ...];</pre>	<p>Use this option to list the StreetTalk servers in order of preference.</p>

<pre>option subnet-mask <i>ip-address</i>;</pre>	<p>Use this option indicate the client's subnet mask as per RFC 950. If no subnet mask option is in scope, the DHCP server uses the subnet mask from the subnet declaration on which the address is being assigned. If a subnet mask option is in scope for the address being assigned, it overrides the subnet mask specified in the subnet declaration.</p>
<pre>option swap-server <i>ip-address</i>;</pre>	<p>Use this option to identify the IP address of the client's swap server.</p>
<pre>option tcp-keepalive-garbage <i>flag</i>;</pre>	<p>Use this option to indicate whether the client sends TCP keepalive messages with an octet of garbage for compatibility with older implementations.</p> <p>ON means do not send a garbage octet. OFF means send a garbage octet.</p>
<pre>option tcp-keepalive-interval <i>uint32</i>;</pre>	<p>Use this option to indicate the interval (in seconds) the client TCP waits before sending a keepalive message on a TCP connection. The time is specified as a 32-bit unsigned integer.</p> <p>0 (zero) means do not generate keepalive messages unless requested by an application.</p>
<pre>option tftp-server-name <i>string</i>;</pre>	<p>Use this option to identify a TFTP server. If this option is supported by the client, it should have the same effect as the <code>server-name</code> declaration. BOOTP clients are unlikely to support this option. Some DHCP clients support it; others require it.</p>
<pre>option time-offset <i>int32</i>;</pre>	<p>Use this option to specify the offset of the client's subnet (in seconds) from Coordinated Universal Time (UTC). Use negative numbers for West of UTC and positive number for East of UTC.</p>



<pre>option time-servers ip-address [, ip-address ...];</pre>	Use this option to list RFC 868 time servers in order of preference.
<pre>option trailer-encapsulation flag;</pre>	<p>Use this option to indicate if the client negotiates the use of trailers (RFC 893) when using the ARP protocol.</p> <p>ON means do not use trailers. OFF means use trailers.</p>
<pre>option vendor-encapsulated- options data-string;</pre>	Use this option to specify vendor specific information. See the <i>Vendor Encapsulated Options</i> section.
<pre>option www-server ip-address [, ip-address ...];</pre>	Use this option to list WWW servers in order of preference.
<pre>option x-display-manager ip-address [, ip-address ...];</pre>	Use this option to list the systems running X Window System Display Manager in order of preference.

## Relay Agent Information Option

A relay agent can add a series of encapsulated options to a DHCP packet when relaying that packet to the DHCP server. The server can make address allocation decisions (or whatever decisions it wants) based on these options. The server returns these options in any replies it sends through the relay agent. The relay agent can use the information in these options for delivery or accounting purposes.

The relay agent option has two suboptions. To reference these options in the DHCP server, specify the option space name `agent`, followed by a period, followed by the option name.

**Note:** It is not useful to specify these options to be sent.

<pre>option agent.circuit-id string;</pre>	<p>The <code>circuit-id</code> suboption encodes an agent-local identifier of the circuit from which a DHCP client-to-server packet was received. It is intended for agents who will use it in relaying DHCP responses back to the proper circuit. The format of this option is defined to be vendor-dependent.</p>
<pre>option agent.remote-id string;</pre>	<p>The <code>remote-id</code> suboption encodes information about the remote host end of a circuit. Examples include the following: caller ID information, username information, remote ATM address, and cable modem ID. This is an opaque object that is administratively guaranteed to be unique to a particular remote end of a circuit.</p>

## Defining New Options

You can define new options with the DHCP server. Each DHCP option has the following:

- A **name**, used by you to refer to the option.
- A **code**, a number used by the DHCP server to refer to the option.
- A **structure**, describing what the contents of the option look like.

To define a new option, choose a **name** that is not in use for any other option. For example, you cannot use `host-name` because the DHCP protocol already defines a `host-name` option. You should refer to the options listed in this chapter as these are all the DHCP options in use by MultiNet.

After choosing a name, choose a **code**. For site-local options, all codes between 128 and 254 are reserved for DHCP options, so you can use any one of these.

The **structure** of an option is the format in which the option data appears. The DHCP server supports a few simple types: for example, integers, booleans, strings, and IP addresses. The server also supports the ability to define arrays of single types or arrays of fixed sequences of types. The syntax for declaring new options is:

```
option new-name code new-code = definition ;
```

The values of `new-name` and `new-code` are the name and the code you have chosen for the new option. The `definition` is one of the following simple option type definitions.

BOOLEAN	<pre>option new-name code new-code = boolean ;</pre>
---------	--

	<p>An option of type <code>boolean</code> is a flag with a value of either <code>ON</code> (true) or <code>OFF</code> (false). For example:</p> <pre>option use-zephyr code 180 = boolean; option use-zephyr on;</pre>
INTEGER	<pre>option new-name code new-code = sign integer width ;</pre> <p>The <code>sign</code> token should either be blank, <code>unsigned</code>, or <code>signed</code>. The width can be either 8, 16 or 32, referring to the number of bits in the integer. For example, a definition of the <code>sql-connection-max</code> option and its use:</p> <pre>option sql-connection-max code 192 = unsigned integer 16; option sql-connection-max 1536;</pre>
IP- ADDRESS	<pre>option new-name code new-code = ip-address ;</pre> <p>An option of type <code>ip-address</code> can be expressed either as a domain name or as an explicit IP address. For example:</p> <pre>option sql-server-address code 193 = ip-address; option sql-server-address sql.example.com;</pre>
TEXT	<pre>option new-name code new-code = text ;</pre> <p>An option of type <code>text</code> encodes an ASCII text string. For example:</p> <pre>option sql-default-connection-name code 194 = text; option sql-default-connection-name "PRODZA";</pre>
DATA STRING	<pre>option new-name code new-code = string ;</pre>

	<p>An option of type <code>string</code> is a collection of bytes. It can be specified either as quoted text, like the <code>text</code> type, or as a list of hexadecimal octets separated by colons whose values must be between 0 and FF. For example:</p> <pre>option sql-identification-token code 195 = string; option sql-identification-token 17:23:19:a6:42:ea:99:7c:22;</pre>
ARRAYS	<p>Options can contain arrays of any of the above types except for the <code>text</code> and the <code>string</code> types. For example:</p> <pre>option kerberos-servers code 200 = array of ip-address; option kerberos-servers 10.20.10.1, 10.20.11.1;</pre>
RECORDS	<p>Options can contain data structures consisting of a sequence of data types, sometimes called a record type. For example:</p> <pre>option contrived-001 code 201 = { boolean, integer 32, text }; option contrived-001 on 1772 "contrivance";</pre> <p>It is also possible to have options that are arrays of records. For example:</p> <pre>option new-static-routes code 201 = array of {     ip-address, ip-address, ip-address, integer 8 }; option static-routes     10.0.0.0 255.255.255.0 net-0-rtr.example.com 1,     10.0.1.0 255.255.255.0 net-1-rtr.example.com 1,     10.2.0.0 255.255.224.0 net-2-0-rtr.example.com 3;</pre>

## Vendor Encapsulated Options

The DHCP protocol defines the `vendor-encapsulated-options` option. This allows vendors to define their own options that will be sent encapsulated in a standard DHCP option. The format of the `vendor-encapsulated-options` option is either a chunk of opaque data, or an actual option buffer just like a standard DHCP option buffer.

You can send this option to clients in one of two ways:

- define the data directly, using a text string or a colon-separated list of hexadecimal values
- define an option space, define some options in that option space, provide values for them, and specify that this option space should be used to generate the `vendor-encapsulated-options` option

To send a simple chunk of data, provide a value for the option in the right scope. For example:

```
option vendor-encapsulated-options
  2:4:AC:11:41:1:
  3:12:73:75:6e:64:68:63:70:2d:73:65:72:76:65:72:31:37:2d:31:
  4:12:2f:65:78:70:6f:72:74:2f:72:6f:6f:74:2f:69:38:36:70:63;
```

To define a new option space to store vendor options, use the `option space` statement. The name can then be used in option definitions. For example:

```
option space SUNW;
option SUNW.server-address code 2 = ip-address;
option SUNW.server-name code 3 = text;
option SUNW.root-path code 4 = text;
```

Once you have defined an option space and some options, you can set up scopes that define values for those options and when to use them. For example, suppose you want to handle two different classes of clients. Using the option space definition, the previous `option vendor-encapsulated-options` example can be implemented using classes as follows:

```
class "vendor-classes" {
  match option vendor-class-identifier;
}

option SUNW.server-address 172.17.65.1;
option SUNW.server-name "sundhcp-server17-1";

subclass "vendor-classes" "SUNW.Ultra-5_10" {
  vendor-option-space SUNW;
  option SUNW.root-path "/export/root/sparc";
}

subclass "vendor-classes" "SUNW.i86pc" {
  vendor-option-space SUNW;
  option SUNW.root-path "/export/root/i86pc";
}
```

Regular scoping rules apply. This lets you define values that are global in the global scope, and define values that are specific to a particular class in the local scope.

The `vendor-option-space` declaration indicates that in that scope the `vendor-encapsulated-options` option should be constructed using the values of all the options in the `SUNW` option space.

# DHCP Lease Format

The DHCP server keeps a persistent database of leases it has assigned. This database is a free-form ASCII file containing a series of lease declarations. Every time a lease is acquired, renewed, or released, its new value is recorded at the end of the lease file. So, if more than one declaration appears for a given lease, the last one in the file is the current one.

Currently, the only declaration that is used in the `dhcpd.leases` file is the `lease` declaration.

```
lease ip-address {statements...}
```

Each lease declaration includes the client's leased IP address. The statements within the braces define, for example, the duration of the lease and to whom it is assigned.

The below table describes the statements the DHCP server puts into a lease file

Lease Statement	Description
<code>abandoned;</code>	Records that the DHCP server saw the IP address in use on the network when it was thought to be free. The DHCP server detects active addresses with ping tests or "DHCP decline" messages from DHCP clients.
<code>billing class "class-name";</code>	If this lease is a member of a class with "lease limit" set, this records that class.
<code>billing subclass "class-name" subclass-data;</code>	If this lease is a member of a subclass with "lease limit" set, this records the class and subclass.
<code>client-hostname "hostname";</code>	Records the hostname if the client sends a hostname using the hostname option.
<code>domain-name "domain-name";</code>	Specifies the DNS domain name sent to the client (if any).
<code>dynamic-bootp;</code>	Indicates the address was leased to a BOOTP client.
<code>ends date;</code>	Records the end time of a lease.  Lease dates are specified by the DHCP server as follows:

	<p><i>W YYYY/MM/DD HH:MM:SS</i></p> <p><i>W</i> is the day of the week, from zero (Sunday) to six (Saturday).  <i>YYYY</i> is the year, including the century.  <i>MM</i> is the number of the month, from 01 to 12.  <i>DD</i> is the day of the month, counting from 01.  <i>HH</i> is the hour, from 00 to 23.  <i>MM</i> is the minute, from 00 to 59.  <i>SS</i> is the second, from 00 to 59.</p> <p>The time is always in Greenwich Mean Time, not local time.</p>
<p>FQDN  <code>"fully-qualified-domain-name";</code></p>	<p>Specifies the fully qualified domain name used by the DHCP server to perform the dynamic DNS update for the lease (if any).</p>
<p>hardware  <code>hardware-type mac-address;</code></p>	<p>Specifies the hardware type and the MAC address as a series of hexadecimal octets, separated by colons.</p>
<p>hostname  <code>"hostname";</code></p>	<p>Records the hostname if the DHCP server looks up the hostname in DNS. This happens only if the parameter <code>get-lease-hostnames</code> was set.</p>
<p><code>starts date;</code></p>	<p>Records the start time of a lease.</p>
<p>uid <code>client-identifier;</code></p>	<p>Records the client identifier as a series of hexadecimal octets, regardless of whether the client specifies an ASCII string or uses the hardware type/MAC address format. If the client used a client identifier to acquire its address, the client identifier is recorded using the <code>uid</code> statement.</p>

## Working with DHCP Leases

The DHCP server requires that a lease database be present before it will start. The MultiNet installation supplies an empty `MULTINET:DHCPD.LEASES` file.

In order to prevent the lease database from growing without bound, the file is rewritten from time to time. First, a temporary lease database is created and all known leases are dumped to it. Then, the old

lease database is renamed `MULTINET:DHCPD.LEASES_OLD`. Finally, the newly written lease database is moved into place.

Be aware of the following situation: if the DHCP server process is killed or the system crashes after the old lease database has been renamed but before the new lease database has been moved into place, the `MULTINET:DHCPD.LEASES` file disappears. The DHCP server will refuse to start. Do not create a new lease file when this happens. If you do, you will lose all your old bindings. Instead, rename `MULTINET:DHCPD.LEASES_OLD` to `MULTINET:DHCPD.LEASES`, restoring the old, valid lease file, and then start the DHCP server. This guarantees that a valid lease file will be restored.

## Abandoned Leases

Abandoned leases are reclaimed automatically. When a client asks for a new address, and the server finds that there are no addresses available, it checks to see if there are any abandoned leases. The server allocates the oldest abandoned lease. The standard procedures for checking for lease address conflicts are still followed, so if the abandoned lease's IP address is still in use, it is re-abandoned.

If a client requests an abandoned address, the server assumes that the address was abandoned because the lease file was corrupted, and that the client is the machine that responded when the lease was pinged, causing it to be abandoned. In that case, the address is immediately assigned to the requesting client.

## Static Leases

Leases that are given to clients for statically assigned IP addresses are treated differently than those for dynamically assigned IP addresses. An address is statically assigned by using a `host` declaration with a `fixed-address` statement.

Static lease information is not saved by the DHCP server. This means that they are not recorded in the lease file (`DHCPD.LEASES`). If your configuration uses only statically assigned IP addresses, you will not see any entries in the lease file.

This also means that the `NETCONTROL SHOW` commands do not have any lease information to display for static assignments.

- For `SHOW CLIENT`, statically assigned IP addresses are not supported.
- For `SHOW SUBNET` and `SHOW LEASES`, statically assigned IP addresses are not shown.



- For `SHOW ALL`, `SHOW HADDR`, `SHOW CID`, and in the dump file produced by the `DUMP` command, statically assigned IP addresses are identified as "Static Assignment" and no lease information is shown.
- For `STATISTICS` and `SHOW POOLS`, statically assigned IP addresses are not included in the pool or subnet statistics.

DNS dynamic updates are supported only partially for static assignments. When the lease is granted, the appropriate A and PTR resource records are added automatically. However, since the lease information is not saved, the DHCP server does not delete the DNS entries when the lease expires or is released.

## Registering Clients While the DHCP Server is Running

The DHCP server can register and unregister clients without having to restart the server. `host` declarations and `subclass` declarations can be added or removed from the running server using `add` and `remove` commands in an update file, by default `MULTINET:DHCPD.UPDATES`.

The commands that can be placed into the update file are described in the table below.

You would use `host` declarations if you are controlling access to IP addresses via `allow/deny` `unknown-clients` statements in your `DHCPD.CONF` configuration file. You would use `subclass` statements if you are controlling access to IP address pools using classes configured with the `match` statement and using pools with `allow/deny` members of `_class_` statements.

**Note:** The registration or un-registration of a client via the update file only affects the running server. The `host` and `subclass` declarations **must** also be added to the `DHCPD.CONF` configuration file.

You tell the DHCP server to execute the commands in the update file using the `NETCONTROL DHCP UPDATE` command:

```
$ multinet netcontrol dhcp update
```

A different file name can optionally be specified:

```
$ multinet netcontrol dhcp update mydir:dhcpd.updates
```

You can verify the syntax of the update file before sending it to the DHCP server by using the `-u` flag on the `dhcpd` command line:

```
$ dhcpd := $multinet:dhcpd.exe
```

```
$ dhcpd -t -u filename
```

The update file name is not optional. Note that the configuration file is read in before the update file. A different configuration file can be specified using the `-cf` flag.

You can check the DHCP server and see if a given host or subclass is known, for example to see if you need to add it, using the following `netcontrol` commands:

```
$ multinet netcontrol show isknown host hw-addr-or-client-id
```

```
$ multinet netcontrol show isknown subclass class-name subclass-data
```

## Update File Statements

The below table describes the commands you can use in an update file.

Statement	Description
add host	<p>Registers a client by adding the specified host declaration. The host declaration is in the same format as in the configuration file. This makes the specified hardware address and/or client identifier "known".</p> <pre>add host name { [statements] }</pre> <p>Note that static IP address assignments can be added by specifying the <code>fixed-address</code> statement in the host declaration.</p>
add subclass	<p>Registers a client by adding the specified subclass to the specified class. The class must be declared in the <code>DHCPD.CONF</code> configuration file. The subclass declaration is the same format as in the configuration file. This adds the specified subclass value as a member of the specified class.</p> <pre>add subclass "class-name" subclass-data; add subclass "class-name" subclass-data {</pre>

	<pre>[statements] }</pre>
delete host	<p>Un-registers a client by removing the specified host declaration. The host is specified by hardware address or client identifier. This makes the specified host "unknown". Note that all host declarations that match the hardware address or client identifier are deleted.</p> <pre>delete host hw-addr-or-client-id;</pre>
delete subclass	<p>Un-registers a client by removing the specified subclass from the specified class. This makes the specified subclass no longer a member of the class.</p> <pre>delete subclass "class-name" subclass-data;</pre>

### Examples:

```
add host fred {
  hardware ethernet 01:02:03:04:05:06;
  fixed-address 10.9.8.7;
  option routers 10.9.8.1;
}
add host wilma {
  option dhcp-client-identifier "wilma-cid";
}
delete host 01:02:03:04:05:06;
delete host "wilma-cid";

add subclass "gold" 01:01:02:03:04:05:06 {
  option host-name "fred";
}
add subclass "silver" "wilma-cid";
delete subclass "gold" 01:01:02:03:04:05:06;
delete subclass "silver" "wilma-cid";
```

# DHCP Safe-failover Introduction

Since a DHCP server is responsible for the network's IP management, it can also be a potential point of network failure if it becomes unavailable. Using multiple servers with non-overlapping IP address pools is one way to provide limited fault-tolerance. For example: imagine a network with two DHCP servers. Server A has an address range of 100 IP addresses. Server B has a range of 50 different addresses. Both servers have a non-overlapping range of addresses. When a node broadcasts for an address, both servers respond, each offering an address from its own distinct range. Upon receiving both offers, the node chooses one. Typically, the response that reaches the node first is selected. In this case, Server A's. When Server B determines its offer is not needed, it returns the offered address to its own range, allowing it to be offered again.

If one of the servers is down, the other server continues to service the nodes. Now, instead of having two offers, each new node has only one offer, from the remaining server. Nodes that received their lease from the unavailable server attempt to reconnect with it. If the unavailable server does not respond in time, the nodes then attempt to get a new address from a new server. The other server can then offer an address from its own range. So, even though one server is down, the DHCP clients continue to function with the other server.

**Note:** The two DHCP servers operate without any communications or data sharing between them. Each server works as a standalone server.

Since most nodes select the first offer received, having two servers could result in partial use of both IP address pools. Sometimes it is preferable to have a primary DHCP server with the bulk of the IP addresses while the secondary server has a smaller range of IP addresses.

**Note:** One way to accomplish the above-mentioned configuration is to put the secondary server behind a router on a different subnet, while the primary server stays on the same segment as the nodes. This allows the primary server to respond more quickly than the secondary server.

Process Software takes the use of multiple servers to another level by offering DHCP Safe-failover. DHCP Safe-failover allows a secondary DHCP server to back up the primary DHCP server with the

addition of taking into account network failure. This strategy insures that clients can reliably log into their corporate network and know they will be able to connect to corporate resources.

In safe failover mode both the primary and the backup DHCP servers share a common IP address lease pool. In the event the primary DHCP server fails the backup DHCP server automatically senses the primary server is not operating and automatically assumes complete DHCP operation. When the primary DHCP server becomes operational, it synchronizes with the backup DHCP server and resumes all the responsibilities of the primary DHCP server. All assignments performed by the backup DHCP server while acting as the primary server are transferred to the primary DHCP upon resumption of primary server responsibilities.

Safe-failover adds support for network failure, as well as server failure. In the event the network fails, the secondary server believes the primary server is out of service and begins operation. The secondary server serves leases from a reserved pool shared by the Safe-failover partner servers. This reserve pool can be adjusted by the MIS system administrator.

## Configuring DHCP Safe-Failover

To configure your DHCP servers to use Safe-failover, perform the following steps:

1. Choose one system to be the Primary and a second system to be the Secondary.
2. Determine the IP addresses of the Primary and Secondary systems. If a system has more than one IP address, choose one to use for DHCP Safe-failover messages.
3. On the Primary system, create the boot file at `MULTINET:DHCPD.BOOT` with the keyword `primary`, the primary and secondary IP addresses, and configuration ID.

On the primary system, the configuration ID would normally be `dhcpd`. See *Boot File for DHCP Safe-failover* for more information about boot files.

Primary system boot file syntax:

```
primary <primary-ip> <secondary-ip> "config-id";
```

4. On the Secondary system, create the boot file at `MULTINET:DHCPD.BOOT` with the keyword `secondary`, the secondary and the primary IP addresses, and configuration ID. On the secondary system, the configuration ID may be `"dhcpd"` or may be a name that refers to the primary. Either way, the names of the configuration, lease, and state files must match this name.

Secondary system boot file syntax:

```
secondary <secondary-ip> <primary-ip> "config-id";
```

5. If you don't already have a configuration file, write a configuration file containing the subnets, shared networks, IP address ranges, hosts, etc, that reflect your network and the hosts you want the DHCP server to service. Include any DHCP Safe-failover parameters as needed (see *DHCP Safe-failover Configuration File Statements*).

6. Copy the configuration file to the `MULTINET` directory on both the Primary and the Secondary systems.

**Note:** Make sure the name of the configuration file matches the `config-id` parameter in the boot file for that system.

Preferably, the configuration files on the Primary and the Secondary server systems should be the same. To help ensure that the configuration file is valid for both systems, make sure it contains a `subnet` statement for every subnet that either the Primary or the Secondary system has a network interface on.

7. Make sure that both the Primary and the Secondary systems have lease files in the `MULTINET` directory with the name that matches `config-id`. If the lease file does not exist, create an empty one.

8. Run the DHCP server on both the Primary and the Secondary systems. The two servers will establish communications with each other and you're in business!

## Boot File for DHCP Safe-Failover

To use Safe-failover, create a boot file at `MULTINET:DHCPD.BOOT`. The boot file is used to specify the following for Safe-failover operation:

- Server's mode of operation
- Server's own IP address
- Partner server's IP address
- Configuration ID

The format of the boot file is:

```
mode [server-IP partner-IP] "config-id";
```

If the boot file is not present, the server operates with DHCP Safe-failover disabled. It uses `multinet:dhcpd.conf` and `multinet:dhcpd.leases` for its default configuration and lease files. In this state, the service parameters `CONFIGFILE` and `LEASEFILE` may be used to rename or move these files. The server does not use a state file to keep track and remember its state transitions when running in standalone mode (that is, with DHCP Safe-failover disabled). See *State File for DHCP Safe-failover* for a description of state files. The following server modes are possible:

Mode	In this mode the server runs...
Primary	As the Primary server, with DHCP Safe-failover functionality enabled. In this mode, the server tries to "shadow" each of its lease transactions to its configured secondary server.
Secondary	As the Secondary or Backup server, with DHCP Safe-failover functionality enabled. The server receives lease transaction updates from its configured Primary server, and maintains an up-to-the-minute hot backup of the lease database. If the primary server crashes, or is shut down, the Secondary server takes over.
Standalone	With DHCP Safe-failover disabled.

The IP address following the server mode is the server's own address, the next IP address is the partner server's IP address.

The configuration ID is the file name (without the file type) of the configuration, lease, and state files. For example, if the configuration id is `ALPHA`, the DHCP server will look for a configuration file named `ALPHA.CONF`, a lease file named `ALPHA.LEASES`, and a state file named `ALPHA.STATE`.

Below is an example boot file:

```
primary 10.23.24.11 10.23.24.25 "ALPHA";
```

The example boot file designates the current server as the primary with its own IP address 10.23.24.11 and the partner server's IP address 10.23.24.25. The partner server is a Secondary server. This follows from the current server being a Primary server.

If the mode of operation is "standalone", the server's IP address and partner server's IP address are not needed. The format is as follows:

```
standalone "config-id";
```

# State File for DHCP Safe-Failover

The state file is written by the DHCP server when it is running with DHCP Safe-failover enabled. The purpose of the state file is to save server state changes so that a server can "remember" its last state if it crashes or is re-started. Alternately, the state file can be used by the operator to force the DHCP server to start up in a desired mode (operator override). This feature allows the operator to switch the server into partner-down mode without a lengthy time-out period, or to start up in recover mode (that is, to force the DHCP server to download the entire lease database from the partner).

The server appends a line to the state file every time its DHCP Safe-failover state changes. The last line in the file is the most current DHCP Safe-failover state.

The state file consists of one or more lines of the following format:

```
server_state transaction_count; [date-time;]
```

*server\_state* is one of the following:

failover-disabled	primary-comint	primary-conflict
startup	backup-comint	backup-conflict
primary-normal	primary-partnerdown	primary-recover
backup-normal	backup-partnerdown	backup-recover

Server-to-server messages are each assigned a monotonously increasing transaction number, which is recorded in the *transaction\_count* field. This is an unsigned 32 bit number.

If the date-time stamp is present, the DHCP server assumes that the state was recorded by the server itself. In this case, the server, upon starting up, calculates the safest state based on the recorded state and the time elapsed. If the date-time stamp is not present, the DHCP server treats the entry as an operator-written override command and starts up in the specified state.

## DHCP Safe-Failover Configuration File Statements

The statements shown in the below table have been added to the DHCP configuration file for DHCP Safe-failover. These are in addition to the DHCP configuration file statements listed in *DHCP Option Space Options*. All of the parameters in *DHCP Server Parameters* must be placed in the configuration



file's global scope. With the exception of the backup-pool-size parameter. It may also be specified within a shared-network or subnet declaration.

Statement	Description
backup-ack-interval	<p>The number of seconds used as the basis of the DHCP server's logarithmic back-off algorithm used for resending ACK messages to the secondary server. The default is 1 second.</p> <pre>backup-ack-interval seconds;</pre>
backup-pool-size	<p>This is the percentage of the IP address pool set aside for the Secondary server's emergency use. The DHCP server will reserve no more than 50% of the available addresses. The default is 10%.</p> <pre>backup-pool-size percent;</pre>
com-int-timeout	<p>The number of seconds the server should wait for an expected response from its partner before switching to communication interrupted mode. The default is 600 seconds (10 minutes).</p> <pre>com-int-timeout seconds;</pre>
failover-port	<p>The UDP port the Primary and Secondary servers should use for DHCP Safe-failover messages. The default is 647.</p> <pre>failover-port port;</pre>
mclt	<p>Maximum Client Lead Time: This is the length of lease in seconds to give out on a first time basis, or the client lead time for renewals. See the DHCP failover internet draft for a more detailed explanation. The default is 3600 seconds (1 hour).</p> <pre>mclt seconds;</pre>

safe-period-timeout	<p>The number of seconds spent in communication interrupted state before automatic switch over to partner-down state. A value of 0 means no automatic switch over. The default is 0 seconds.</p> <pre>safe-period-timeout seconds;</pre>
startup-delay	<p>The number of seconds to wait during startup before the server moves from STARTUP state to the state specified in the state file. The default is 5 seconds.</p> <pre>startup-delay seconds;</pre>

## DHCP Safe-Failover Lease File Statements

The statements shown in the below table have been added to the DHCP lease file for DHCP Safe-failover. These are in addition to the DHCP lease file statements listed in *DHCP Lease File Statements*.

Statement	Description
<code>acked-sec-interval seconds;</code>	Acknowledged secondary lease interval. For information, see the DHCP failover internet draft.
<code>acked-sec-interval-start date;</code>	The time when the partner was notified of the lease.
<code>active;</code>	This IP address has a current lease.
<code>backup;</code>	This IP address is reserved for use by the secondary (backup) server.
<code>desired-interval seconds;</code>	The length of the desired lease.

<code>expired;</code>	The lease for this IP address has expired.
<code>free;</code>	This IP address is available to be assigned.
<code>last-partner-transaction date;</code>	The time when the partner last updated the lease.
<code>released;</code>	The lease for this IP address has been released by the client or by the operator.
<code>reset;</code>	The DHCP server had marked this IP address as abandoned. The operator changed its status to available.
<code>revoked;</code>	The lease for this IP address has been revoked by the operator.
<code>safe-lease;</code>	This is used in the Partner Down state to indicate that the IP address belongs to this server.
<code>transaction-id <i>number</i>;</code>	This is the transaction number that was assigned to this lease when it was queued or sent as an update to the partner server.
<code>update-count <i>n</i>;</code>	<p>For each lease, the server which issues the lease sends an update to its partner server. This statement records the state of that update.</p> <p>0 - means no update is required</p> <p>1 - means that no update has been sent</p> <p>2+ - means 1 or more updates have been sent</p>

# Transitioning to DHCP Safe-Failover Partner Down State

There are three ways that the DHCP server can transition to Partner Down state, which indicates that its DHCP Safe-failover partner is down.

1. If the parameter `safe-period-timeout` is specified in the configuration file, the DHCP server transitions to Partner Down state automatically after it has been in Communication Interrupt state for the specified time.
2. The operator can put the DHCP server into Partner Down state by executing the following `netcontrol` command:

```
$ multinet netcontrol dhcp partnerdown
```

3. The operator can edit the DHCP server's state file and add a line to the end containing the Partner Down state and transaction number desired:

```
primary-partnerdown transaction-number;
```

or

```
backup-partnerdown transaction-number;
```

The next time the DHCP server is restarted, it starts up in Partner Down state. The operator can restart the DHCP server by executing the following `netcontrol` command:

```
$ multinet netcontrol dhcp restart
```

## Setting DHCP Parameters

The DHCP service uses the parameters listed in the table below.

Parameter	Description
ACCOUNTING	This parameter determines if the DHCP server process performs VMS accounting. The default is 0 (OFF).
CONFIGFILE	The name of the configuration file. The default is <code>MULTINET:DHCPD.CONF</code> . Not used if DHCP Safe-failover is being used.

DEBUG	<p>A decimal integer that is a bitmask of debugging levels used to select messages to pass to OPCOM and the debug log file specified in the DEBUG-FILE parameter. The debugging levels are (in decimal):</p> <ul style="list-style-type: none"> <li>1 Fatal Errors</li> <li>3 Errors and Warnings</li> <li>7 Informationals</li> <li>15 Debug Messages</li> <li>31 Dump Packets (Formatted)</li> <li>63 Dump Packets (Hex)</li> </ul> <p>By default, Fatal Errors, Errors, and Warnings are logged.</p>
DEBUG-FILE	<p>The name of the debug log file. Use this parameter to capture debug logging in a file. The default is that debug logging is not written to any file if LOG-TO-OPCOM is 1. If LOG-TO-OPCOM is 0, the default file is MULTINET : DHCPDEBUG . LOG</p>
DUMPFIL	<p>The name of the "dump" file. This is the output of the NETCONTROL DUMP command. The default is MULTINET : DHCPD . DUMP.</p>
IMAGE-NAME	<p>The name of the image to run in the DHCP server process. The default is MULTINET : DHCPD . EXE.</p>
LEASEFILE	<p>The name of the file DHCP uses to save client lease information to survive across reboots. To completely clear the lease information, delete the file specified and create an empty version. The default is MULTINET : DHCPD . LEASES.</p> <p>Not used if DHCP Safe-failover is being used.</p>
LOG-DATE	<p>This parameter determines whether the date is included in each entry in the debug log file. The default is 0; the date is not included.</p>
LOG-TO-OPCOM	<p>This parameter determines whether debug logging messages are written to OPCOM. Debug messages are also written to DEBUG-FILE parameter value if DEBUG-</p>

	FILE is specified or this parameter is 0. Severe errors and warnings are always logged to OPCOM. The default is 1, everything is logged to OPCOM.
PROCESS-NAME	The name of the DHCP server process. The default is DHCP_SERVER.
SYS-ERROR	The name of a file that will contain anything written to SYS\$ERROR by the DHCP server. This information could be helpful in diagnosing problems. The default is _NL:. This means SYS\$ERROR is not directed to any file.
SYS-OUTPUT	The name of a file that will contain anything written to SYS\$OUTPUT by the DHCP server. This information could be helpful in diagnosing problems. The default is _NL: meaning SYS\$OUTPUT not directed to any file.
SWAP	This parameter determines whether process swapping is inhibited for the DHCP server process. The default is 1, swapping is enabled.
UPDATEFILE	The name of the update file. This file contains update commands that the DHCP server executes upon a NETCONTROL UPDATE command. The default is MULTINET:DHCPD.UPDATES.

You may set any of the parameters listed in *DHCP Lease File Statements*, as shown in the following example:

```
$ MULTINET NETCONTROL DHCP SHUTDOWN (if DHCP is running)
$ MULTINET CONFIGURE /SERVER
MultiNet Server Configuration Utility 5.6
[Reading in configuration from MULTINET:SERVICES.MASTER_SERVER]
SERVER-CONFIG>SELECT DHCP
[The Selected SERVER entry is now DHCP]
SERVER-CONFIG>SET PARAMETERS
Delete parameter "configfile MULTINET:DHCPD.CONF" ? [NO] RETURN
You can now add new parameters for DHCP. An empty line terminates.
Add Parameter: debug 3
Add Parameter: RETURN
[Service specific parameters for DHCP changed]
SERVER-CONFIG>RESTART
Configuration modified, do you want to save it first ? [YES] RETURN
[Writing configuration to MULTINET_COMMON_ROOT:[MULTINET]
SERVICES.MASTER_SERVER]
%RUN-S-PROC_ID, identification of created process is 20600046
SERVER-CONFIG>EXIT
```

```
[Configuration not modified, so no update needed]
```

```
$
```

## Viewing DHCP Information

MultiNet provides two NETCONTROL commands for displaying information about the DHCP server:

- NETCONTROL SHOW (see the *NETCONTROL SHOW Command* section)
- NETCONTROL STATISTICS (see the *NETCONTROL STATISTICS Command* section)

### NETCONTROL SHOW Command

The DHCP NETCONTROL SHOW command has seven subcommands: SHOW CLIENT, SHOW LEASES, SHOW SUBNET, SHOW ALL, SHOW HADDR, SHOW CID, and SHOW POOLS.

#### Viewing Lease Information for Specific IP Addresses

The DHCP NETCONTROL SHOW CLIENT command displays the current lease binding details on a particular IP address. It must be an IP address in the dynamic pool. Statically-bound IP addresses are not supported. The syntax for SHOW CLIENT is:

```
$ MULTINET NETCONTROL DHCP SHOW CLIENT dotted-decimal-ip-address
```

*dotted-decimal-ip-address* is the IP address of a client.

For example:

```
$ MULTINET NETCONTROL DHCP SHOW CLIENT 10.5.64.1
```

```
Connected to NETCONTROL server on "LOCALHOST"
```

```
< x.process.com Network Control V5.6 at Fri 7-Aug-2019 3:23PM-EDT
```

```
< DHCP Client: 10.5.64.1
```

```
< IP Address=10.5.64.1
```

```
< State=Bound (expired)
```

```
< Subnet Mask=255.255.255.0
```

```
< Default Gateway=10.5.64.100
```

```
< Hardware Address=00004400AABB
```

```
< Client ID=74657374 ("test")
```

```
< Lease=300 secs, Obtained 06-Aug-2019 22:21:22 GMT Expires 16-Aug-2019  
22:26:22 GMT (-75426 secs)
```

```
< End of Show DHCP Client
```

## Viewing Lease Information for all Leased IP Addresses

The DHCP NETCONTROL SHOW LEASES command takes no arguments. It displays for all subnets the IP addresses that have leases (pending, active, or expired). Lease information for statically-assigned IP addresses is not available. For example:

```
$ MULTINET NETCONTROL DHCP SHOW LEASES
Connected to NETCONTROL server on "LOCALHOST"
< x.process.com Network Control V5.6 at Wed 2-Aug-2019 2:32PM-EST
< List all leases
< Shared Network local
< Subnet 10.5.64.0 netmask 255.255.255.0
< Subnet 10.5.165.0 netmask 255.255.255.0
< Pool 1
<   IP Addr=10.5.64.1, State=Bound (expired), Lease Expires
03-Mar-2003 19:32:43 GMT (-4 secs)
<   IP Addr=10.5.64.13, State=Offered, Lease Expires 08-Aug-2019 19:34:07
GMT (80 secs)
<   IP Addr=10.5.64.2, State=Bound, Lease Expires 08-Aug-2010 19:36:52 GMT
(245 secs)
< End of lease list
```

## Viewing Address Pools for Specific Subnets

The DHCP NETCONTROL SHOW SUBNET command displays all of the DHCP address pools for the shared network that ip-address is in. It lists each subnet that is on the shared network and each IP address in each pool. Statically-bound IP addresses are not shown. The syntax for SHOW SUBNET is:

```
$ MULTINET NETCONTROL DHCP SHOW SUBNET dotted-decimal-ip-address
dotted-decimal-ip-address is any IP address in that subnet.
```

You would typically use the subnet value or an IP address from a pool for the subnet. For example:

```
$ MULTINET NETCONTROL DHCP SHOW SUBNET 10.5.64.0
Connected to NETCONTROL server on "LOCALHOST"
< x.process.com Network Control V5.6 at Mon 14-Aug-2019 7:09PM-EDT
< List given Subnet pool
< Shared Network local
< Subnet 10.5.64.0 netmask 255.255.255.0
< Subnet 10.5.165.0 netmask 255.255.255.0
< Pool 1
<   IP Addr=10.5.64.3, State=Free, No Lease
<   IP Addr=10.5.64.1, State=Bound, Lease Expires 14-Aug-2019 22:21:14 EDT
(2867 secs)
< End of Subnet pool list
```



## Viewing Address Pools for All Subnets

The DHCP NETCONTROL SHOW ALL command takes no arguments. It shows the SHOW SUBNET output for all subnets in the DHCP server configuration. Then it prints information about all static assignments.

**Note:** For static assignments, lease information is not available.

For example:

```
$ MULTINET NETCONTROL DHCP SHOW ALL
Connected to NETCONTROL server on "LOCALHOST"
< x.example.com Network Control V5.6 at Tue 11-Jul-2019 11:24AM-EDT
< List all Subnet pools
< Shared Network local
< Subnet 10.5.64.0 netmask 255.255.255.0
< Subnet 10.5.165.0 netmask 255.255.255.0
< Pool 1
<   IP Addr=10.5.64.3, State=Free, No Lease
<   IP Addr=10.5.64.1, State=Bound, Lease Expires 15-Jul-2019 22:21:14 EDT
(2867 secs)
< DHCP Static Assignments
<   IP Addr=10.5.64.15, State=(Static Assignment)
<   IP Addr=10.5.165.17, State=(Static Assignment)
<   IP Addr=10.5.165.200, State=(Static Assignment)
< End of Subnet pool lists
```

## Viewing Matched Leases for Hardware Addresses

The DHCP NETCONTROL SHOW HADDR command shows all client entries that match a given hardware address. The clients can have leases on multiple subnets simultaneously.

The syntax for the NETCONTROL SHOW HADDR command is

```
$ MULTINET NETCONTROL DHCP SHOW HADDR MAC address
```

For example:

```
$ MULTINET NETCONTROL DHCP SHOW HADDR 00004400AABB
Connected to NETCONTROL server on "LOCALHOST"
< x.example.com Network Control V5.6 at Fri 04-Aug-2019 4:41PM-EDT
< DHCP Hardware Address: 00004400AABB
<   IP Address=10.5.64.1
<   State=Bound (expired)
```

```

< Subnet Mask=255.255.255.0
< Default Gateway=10.5.64.100
< Hardware Address=00004400AABB
< Client ID=74657374 ("test")
< Lease=300 secs, Obtained 08-Aug-2019 22:21:22 EDT Expires 08-Aug-2019
22:26:22 EDT (-80091 secs)
< End of Show DHCP HAddr

```

## Viewing Matched Leases for Client ID

The DHCP NETCONTROL SHOW CID command shows all client entries that match a given client ID. The clients can have leases on multiple subnets simultaneously.

The syntax for the NETCONTROL SHOW CID command is

```
$ MULTINET NETCONTROL DHCP SHOW CID Client_ID_in_hex
```

For example:

```

$ MULTINET NETCONTROL DHCP SHOW CID 7465737431
Connected to NETCONTROL server on "LOCALHOST"
< x.example.com Network Control V5.6 at Mon 10-Jul-2019 6:36PM-EDT
< DHCP Client ID: 7465737431
< IP Address=10.24.25.1
< State=(Static Assignment)
< Subnet Mask=255.255.255.0
< Default Gateway=<none>
< Hardware Address=<none>
< Client ID=7465737431 ("test1")
< End of Show DHCP Client ID

```

## Viewing IP Address Pool Availability

The DHCP NETCONTROL SHOW POOLS command takes no arguments. It displays a table showing for each IP address pool the number of IP addresses that are available. An IP address pool corresponds to a shared-network statement, a subnet statement, or a pool statement in the DHCP configuration file. For each pool the following information is displayed:

Shared Network	The name from the shared-network statement or the subnet number from the subnet statement.
Pool	“Total” for the complete information for the shared network, otherwise a number identifying the pool. You can see which IP addresses are in which pools using the SHOW ALL or SHOW SUBNET command.

Total	The total number of IP addresses in the pool.
Abandoned	The number of IP addresses in the pool which were found in use on the network when they were thought to be free.
Reserved	If DHCP Safe-failover is in use, the number of IP addresses in the pool reserved for the secondary DHCP server. These addresses are unassigned but reserved for the secondary.
Available	The number of IP addresses in the pool available to be leased.

For example:

```
$ MULTINET NETCONTROL DHCP SHOW POOLS
Connected to NETCONTROL server on "LOCALHOST"
< x.example.com Network Control V5.6 at Fri 11-Feb-2020 5:25PM-EST
< List Pool Availability
< Shared Network      Pool      Total      Abandoned  Reserved  Available
< -----            ----      -
< local               total    44         5           0         15
<                    1       44         5           0         15
< 10.12.1.0           total    128        2           0         57
<                    1       111        0           0         54
<                    2       11         2           0          0
<                    3        6          0           0          3
< End of pool availability list
```

## NETCONTROL STATISTICS Command

The DHCP NETCONTROL STATISTICS command displays the number of clients in each shared network. This is supplied for backward compatibility only. Use SHOW POOLS instead.

# Address Lease States in DHCP Dump Files

After obtaining a DHCP dump using `MULTINET NETCONTROL DHCP DUMP`, you will see fields in the dump preceded by a pound sign (#). Those fields are values created while the server is running. These fields are provided to help you troubleshoot problems. The lease states (denoted by `# State=` in the dump) are described in the below table.

State	Description
Abandoned	The address was seen in use on the network when it was thought to be free. The DHCP server detects active addresses with ping tests or "DHCP decline" messages from DHCP clients.
Bound	The address was assigned in response to a DHCP Request message. If the address lease expires, it remains in "bound" state to help the client regain the same IP address it previously used. The address is actually free. The "(expired)" modifier on the state value indicates this state.
Free	The address has never been bound, and is available in the pool.
Offered	The address has been offered to a client in response to the client's DHCP Discover message, but the client has not asked for the address via a DHCP Request message.
Pinging	The address is in the middle of a ping test.
Reserved for Secondary	Used for DHCP Safe-failover: the address is set aside for the secondary server's emergency use.
Static Assignment	The client identifier or hardware address is statically assigned; the binding does not expire.

## Sample DHCPD.CONF File

Below is a sample `DHCPD.CONF` file.

```
#
# MULTINET:DHCPD.CONF -- sample DHCP configuration file
#

# option definitions common to all supported networks...
option domain-name "fugue.com";
option domain-name-servers toccato.fugue.com;
default-lease-time 43200;
option subnet-mask 255.255.255.0;
option time-offset 18000;
use-host-decl-names on;

# Shared network declaration is used to group subnets which share the same
# physical network together. The name is specified so that the shared
# network can be referred to in log messages --
# it serves no other function.
#
# Note: You must have a subnet declaration for the subnet that the DHCP
# server system is on even if you don't want any address pool for the same
# subnet (or multiple subnets if the system is multi-homed).

shared-network FUGUE {

# option definitions common to this shared network.
    option subnet-mask 255.255.255.224;
    default-lease-time 600;
    max-lease-time 7200;

# One of the two IP subnets that share this physical network
#
# Address ranges can be specified for each subnet attached to a shared
# network. Since these subnets share the same physical network, addresses
# are pooled together, and assignments are made without regard to the
# actual subnet. If the optional dynamic-bootp keyword is given in the
# address range declaration, then addresses in that range can be assigned
# either with the DHCP protocol or the BOOTP protocol; otherwise, only
# DHCP clients will have addresses allocated from the address range.
#
# Note that each IP subnet can have its own options specific to that
# subnet. Options that are not specified in the subnet are taken from the
# shared network (if any) and then from the global option list.

    subnet 204.254.239.0 netmask 255.255.255.224 {
        range 204.254.239.10 204.254.239.20;
        option broadcast-address 204.254.239.20;
        option routers prelude.fugue.com;
    }

# The other subnet that shares this physical network
    subnet 204.254.239.32 netmask 255.255.255.224 {
        range dynamic-bootp 204.254.239.42 204.254.239.52;
        option broadcast-address 204.254.239.31;
    }
}
```

```

    option routers snarg.fugue.com;
}

# Subnets can have no pooled ip addresses.
  subnet 10.10.10.0 netmask 255.255.255.0 {
  }
}

# IP subnets that are alone on their physical wire should be declared by
# themselves. The DHCP server still refers to them as shared networks in
# log messages, but this is simply an artifact of the underlying data
# structure.
#
# Note that options can be specified in the subnet declaration that
# supercede the global options specified earlier.

subnet 192.5.5.0 netmask 255.255.255.224 {
  range 192.5.5.26 192.5.5.40;
  option domain-name-servers bb.home.vix.com, gw.home.vix.com;
  option domain-name "vix.com";
  option routers 192.5.5.1;
  option subnet-mask 255.255.255.224;
  option broadcast-address 192.5.5.41;
  default-lease-time 600;
  max-lease-time 7200;
}

# Hosts that require special configuration options can be listed in host
# statements. If no address is specified, the address will be allocated
# dynamically (if possible), but the host-specific information will still
# come from the host declaration.

host passacaglia {
  hardware ethernet 0:0:c0:5d:bd:95;
  filename "vmunix.passacaglia";
  server-name "toccato.fugue.com";
}

# Fixed IP addresses can also be specified for hosts. These addresses
# should not also be listed as being available for dynamic assignment.
# Hosts for which fixed IP addresses have been specified can boot using
# BOOTP or DHCP. Hosts for which no fixed address is specified can only be
# booted with DHCP, unless there is an address range on the subnet to
# which a BOOTP client is connected which has the dynamic-bootp flag set.
host fantasia {
  hardware ethernet 08:00:07:26:c0:a5;
  fixed-address fantasia.fugue.com;
}

# If a DHCP or BOOTP client is mobile and might be connected to a variety
# of networks, more than one fixed address for that host can be specified.
# Hosts can have fixed addresses on some networks, but receive dynamically

```

```
# allocated address on other subnets; in order to support this, a host
# declaration for that client must be given which does not have a fixed
# address. If a client should get different parameters depending on what
# subnet it boots on, host declarations for each such network should be
# given. Finally, if a domain name is given for a host's fixed address and
# that domain name evaluates to more than one address, the address
# corresponding to the network to which the client is attached, if any,
# will be assigned.
```

```
host confusia {
    hardware ethernet 02:03:04:05:06:07;
    fixed-address confusia-1.fugue.com, confusia-2.fugue.com;
    filename "vmunix.confusia";
    server-name "toccato.fugue.com";
}
```

```
host confusia {
    hardware ethernet 02:03:04:05:06:07;
    fixed-address confusia-3.fugue.com;
    filename "vmunix.confusia";
    server-name "snarg.fugue.com";
}
```

```
host confusia {
    hardware ethernet 02:03:04:05:06:07;
    filename "vmunix.confusia";
    server-name "bb.home.vix.com";
}
```

```
# Some other examples
```

```
host host1 {
    option dhcp-client-identifier "host1";
    fixed-address 10.10.11.101, 10.11.22.101;
}
```

```
# Do not allow this one to boot
```

```
host host2
    hardware ethernet aa:cc:04:00:33:11;
    deny booting;
}
```

# 19. Managing the XDM Server

This chapter explains how to configure the MultiNet XDM (X Display Manager) server to manage remote systems running X (X Window System) servers. The MultiNet XDM server is based on X11R6.

## Understanding X Display Management

The MultiNet XDM server provides login services to X servers through graphical dialog boxes. From a user's perspective, logging in at a MultiNet XDM-managed X server is no different than logging in at an OpenVMS system console.

Initially, the user sees the standard DECwindows banner and login dialog. After the user supplies a valid user name and password, the DECwindows Session Manager starts and launches any applications configured for automatic startup. After the user logs out from the Session Manager, the banner and login dialog reappear.

Users of X11R4 (and later) servers benefit from XDM because they can take advantage of the DECwindows Session Manager as if they were using a host.

Users can also specify which host they want to use. This feature is provided by XDMCP (X Display Management Communications Protocol), used for communication between X servers and XDM servers.

From a system manager's perspective, XDM provides a convenient way to extend the LOGINOUT service to users on remote X servers and to specify which X servers are managed by each XDM server host.

## Accessing the XDM Server

Before an X server can be managed, you must grant it access to your XDM server. In a network of dozens or hundreds of X servers, it may not be practical to manage every X server that requests XDM



service. In these cases, you can restrict access to your XDM server. The manner in which access is granted depends on the version of the X server and how it requests XDM service.

X11R4 (and later) X servers can generate three basic types of management requests, as explained in the table below.

<b>Request Type</b>	<b>Description</b>	<b>XDM Server Response</b>
Direct	Sent by X server to a specific host	If the XDM server is configured to accept requests from the user's X server, the host produces a login dialog on that server. If the XDM server is configured to refuse service to the user's X server, no login dialog appears, and the user must try a different host. For information on selectively rejecting XDMCP direct requests, see the <i>Handling Direct and Broadcast Requests</i> section.
Broadcast	Broadcast by X server to all hosts on the local network	The XDM server responds to broadcast requests as if they were direct requests. Broadcasts can result in several XDM servers accepting the request; it is then up to the X server to determine which host to use. Some X servers simply use the first host to respond. Others present a list of responding hosts in a chooser dialog box, and the user selects a host from the list.
Indirect	Sent by X server to a specific host	<p>If the host's XDM server is configured to handle indirect requests from the X server, the host either forwards the request to another XDM server host or returns a list of XDM servers from which the user can choose.</p> <div data-bbox="578 1413 1463 1661" style="background-color: #e6f2ff; padding: 10px; border-left: 3px solid #0070c0;"> <p><b>Note:</b> Only X11R5 and X11R6 XDM servers can properly handle forwarded requests.</p> </div> <p><i>The MultiNet XDM server does not handle indirect requests.</i></p>

From the user's perspective, broadcast and indirect requests provide similar benefits. From an administrator's perspective, however, indirect requests allow the XDM server administrator to select an XDM server for users.

Consider a situation in which BROWN.EXAMPLE.COM, an X terminal, is configured to make indirect requests for display management to an OpenVMS host, WHORFIN.EXAMPLE.COM, running the MultiNet XDM server. The administrator of WHORFIN can specify which XDM servers will manage BROWN by forwarding inquiries to another XDM server.

X11R3 servers cannot take advantage of XDMCP, which was not introduced until X11R4. Instead, X11R3 servers do not allow users to choose a host. For information on managing X11R3 servers, see the *Managing X11R3 Displays* section.

## Special Features of the MultiNet XDM Server

Unlike many XDM servers, the MultiNet XDM server does not control X sessions after users have logged in. Instead, it starts the processes that produce the login dialog and authenticate the user, then passes control to the DECwindows Session Manager. When the user terminates the session using the Session Manager, the X session ends and XDM starts a new login cycle. This method allows you to change XDM server configuration, stop the XDM server, and restart the XDM server without interrupting any X sessions already in progress.

Configuration of a remote user's session is identical to the customization of a local DECwindows user's session. Traditionally, XDM servers on UNIX systems allow user environments to be configured by scripts (command procedures) that are run before, during, and upon ending each X session. The MultiNet XDM server, however, takes advantage of the DECwindows STARTLOGIN and LOGINOUT processes to eliminate the need for extra command procedures.

The MultiNet XDM server does not handle indirect requests.

## XDM Administrative Tasks

The following are common administrative tasks for the XDM server:

- Enabling the XDM server (see the *Enabling and Starting the XDM Server* section).
- Modifying the XDM server configuration (see the *Modifying the XDM Server Configuration* section).

- Controlling the XDM server (see the *Controlling the XDM Server* section).
- Controlling access to the XDM server (see the *Controlling Access to the XDM Server* section).
- Managing X11R3 displays (see the *Managing X11R3 Displays* section).

With the exception of controlling the XDM server, each task requires you to modify XDM configuration files and update the XDM server with the new information (as described in the *Controlling the XDM Server* section).

## Enabling and Starting the XDM Server

When MultiNet is first installed, the XDM server is disabled, and you must enable it. After the server is enabled, it starts automatically when MultiNet starts.

1. Issue the following DCL command:

```
$ MULTINET CONFIGURE /SERVER
```

2. At the SERVER-CONFIG prompt, enter:

```
SERVER-CONFIG>ENABLE XDM
```

3. To verify that the XDM server is enabled, enter:

```
SERVER-CONFIG>SHOW XDM
```

If no asterisk (\*) appears to the left of XDM in the output, the service is enabled.

4. To exit the configuration utility, enter:

```
SERVER-CONFIG>EXIT
```

5. Restart the master server:

```
$ @MULTINET:START SERVER.COM
```

## Modifying the XDM Server Configuration

XDM is configured through X resources and files specified through X resources. When the XDM server starts or you reload its configuration, it configures itself according to the contents of its master

configuration file, MULTINET:XDM.CONFIGURATION, including reading the contents of files specified in the master configuration file.

Resource parameters for MultiNet XDM are listed in the below tables.

These resources modify the global behavior of XDM. Because the resource manager uses colons to separate the name of the resource from its value and dots to separate resource name parts, XDM substitutes underscores for both dots and colons when generating the resource name.

**Note:** If you import an XDM configuration file from a UNIX system, the startup, session, and authorize resources have no effect on the MultiNet XDM server.

The following is a sample XDM.CONFIGURATION file:

```
DisplayManager.logFile:      multinet:xdm-errors.log
DisplayManager.servers:     multinet:xdm.servers
DisplayManager.accessFile:  multinet:xdm.access
DisplayManager.removeDomainname: false
```

Configuration File Resource	Description
DisplayManager.accessFile	Specifies a file containing a database of host names that are either allowed direct access to this host or to which queries should be forwarded. For details, see the <i>Controlling Access to the XDM Server</i> section. The equivalent file on UNIX systems is <code>/usr/lib/X11/xdm/Xaccess</code> .  Default: MULTINET:XDM.ACCESS.
DisplayManager.debugLevel	Sets the debug output level. Default: 0.  You can also set this value while the XDM server is running with the command <code>MULTINET NETCONTROL XDM DEBUG</code> .

<code>DisplayManager.logFile</code>	<p>Specifies the file in which errors are logged when the <code>logType</code> resource is <code>FILE</code>. The equivalent file on UNIX systems is <code>/usr/lib/X11/xdm/xdm-errors</code>.</p> <p>Default: <code>MULTINET:XDM.LOG</code>.</p>
<code>DisplayManager.logType</code>	<p>Specifies where XDM server messages are logged: <code>OPCOM</code>, <code>SYSLOG</code>, <code>FILE</code>, or <code>STDOUT</code>.</p> <p>Default: <code>OPCOM</code>.</p>
<code>DisplayManager.removeDomainName</code>	<p>Specifies whether XDM removes the domain name portion of host names if they are the same as the domain name of the local host. Removing the domain name is useful because name resolvers typically create fully qualified host names when computing display names for XDMCP clients.</p> <p>Default: <code>TRUE</code>.</p>
<code>DisplayManager.requestPort</code>	<p>Specifies the UDP port number to which the XDM server listens for incoming XDMCP requests. Unless you need to debug the system, do not change the value from the default.</p> <p>Default: 177 (the standard for XDMCP).</p>
<code>DisplayManager.servers</code>	<p>Specifies a file containing a list of foreign X servers to manage. If the file specification is prefixed with an <code>@</code> sign, it contains one listing per line. A foreign X server is an X display that does not support the XDMCP protocol. The equivalent file on UNIX systems is</p>

	<pre>/usr/lib/X11/xdm/Xservers</pre> <p>Default: MULTINET:XDM.SERVICES.</p>
--	---

You can use the configuration parameters in the below table to adjust the way the XDM server transfers control of the X terminal to a DECwindows process. The default values for these parameters should be suitable for ordinary DECwindows usage.

Under normal operation, the XDM server creates an X connection to the managed display before starting a process to invoke the DECwindows login screen. When the login completes and the DECwindows Session Manager starts, it hands off the initial X connection to the process. The initial X connection closes down automatically when the session ends, causing the X display to reset and either shut down or restart the XDM login chooser (depending on how the X terminal or software is configured).

Configuration Resource	Description
<code>DisplayManager.loginCheckInterval</code>	The number of seconds the XDM server waits between checks to see if the DECwindows login process has begun. Default: 5
<code>DisplayManager.loginTries</code>	The number of checks the XDM server makes to see if the DECwindows login process has begun. Default: 12
<code>DisplayManager.sessionCheckInterval</code>	The number of seconds the XDM server waits between checks to see if the DECwindows login process has handed control to the user's Session Manager. Default: 10
<code>DisplayManager.sessionTries</code>	The number of checks the XDM server makes to see if the user's Session Manager started. Default: 30
<code>DisplayManager.bypassSessionCheck</code>	If set to true, the check for the start of a Session Manager is bypassed. Default: false
<code>DisplayManager.sessionManagers</code>	A comma-separated list of image names, without device or directory specifications, that the XDM server should consider to be Session Managers for a

	DECwindows session. Default: DECW\$SESSION.EXE
--	---

# Controlling the XDM Server

This section describes the following XDM tasks:

- Checking the status of the XDM server
- Starting the XDM server
- Stopping the XDM server
- Restarting the XDM server
- Reloading the XDM configuration

For most of these tasks, you use the `MULTINET NETCONTROL` utility.

## Checking the Status of the XDM Server

To check the status of the MultiNet XDM server, enter:

```
$ MULTINET NETCONTROL XDM SHOW
```

The following example shows the information generated by this command on a system named WHORFIN that manages three X terminals, BANZAI, BROWN, and MIGUEL. When the terminal is displaying a DECwindows login screen, the following command output changes to show the words "not logged in" instead of a process identifier number.

```
$ MULTINET NETCONTROL XDM SHOW
```

```
Connected to NETCONTROL server on "WHORFIN"  
< WHORFIN.EXAMPLE.COM  
< Network Control 5.6 at Wed 8-Oct-2019 11:37AM-EST  
< XDM Current Managed Displays:  
<   Display BANZAI.EXAMPLE.COM:0.0 Type XDMCP Process 20E002A8  
<   Display BROWN.EXAMPLE.COM:0.0 Type XDMCP Process 20E00289  
<   Display MIGUEL.EXAMPLE.COM:2005.0 Type XDMCP Process 20E00242  
< OK
```

# Starting the XDM Server

This section describes how to start the MultiNet XDM server.

**Note:** The XDM server is automatically started when MultiNet starts.

Before starting the MultiNet XDM server, make sure it is enabled. For details, see the *Enabling and Starting the XDM Server* section.

To start the MultiNet XDM server, enter:

```
$ MULTINET NETCONTROL XDM START
```

When the XDM server starts, it reads the master configuration file, `MULTINET:XDM.CONFIGURATION`, to determine which configuration files to read, then reads them. (For information on modifying the master configuration file, see the *Modifying the XDM Server Configuration* section). You can specify other configuration files simply by modifying the contents of the master configuration file and then restarting the XDM server (see *Restarting the XDM Server*), or by reloading the XDM configuration files (see the *Reloading the XDM Configuration* section).

The following example shows the information generated by this command:

```
$ MULTINET NETCONTROL XDM START  
< XDM Server Started, process id pid
```

# Stopping the XDM Server

To stop the MultiNet XDM server, enter:

```
$ MULTINET NETCONTROL XDM SHUTDOWN
```

The following example shows the information generated by this command:

```
$ MULTINET NETCONTROL XDM SHUTDOWN  
< XDM Server Shutdown
```

# Restarting the XDM Server

Restarting the XDM server is a convenient alternative to stopping and starting the XDM server as described in the *Starting the XDM Server* section and the *Stopping the XDM Server* section.



When the XDM server restarts, it reads the master configuration file `MULTINET:XDM.CONFIGURATION` to determine which configuration files to read, then reads them.

To restart the XDM server, enter:

```
$ MULTINET NETCONTROL XDM RESTART
```

The following example shows the information generated by this command:

```
$ MULTINET NETCONTROL XDM RESTART
Connected to NETCONTROL server on "LOCALHOST"
< WHORFIN.EXAMPLE.COM Network Control 5.6 at Tue 30-Sep-2019 12:47AM-EST
< XDM Server Started, process id 2040024B
$
```

For information on modifying the master configuration file, see the *Modifying the XDM Server Configuration* section.

## Reloading the XDM Configuration

Changes to XDM server configuration take effect when the XDM server is started or restarted or when the configuration files are reloaded. Reloading the XDM server allows you to reload the XDM server configuration files without restarting the XDM server and interrupting connections between X servers and the XDM server.

When the XDM server reloads its configuration, it first reads the `MULTINET:XDM.CONFIGURATION` master configuration file to determine which other files to read, then reads them. For information on modifying the master configuration file, see the *Modifying the XDM Server Configuration* section.

To reload the XDM configuration files, enter:

```
$ MULTINET NETCONTROL XDM RELOAD
```

The following example shows the information generated by this command produces:

```
$ MULTINET NETCONTROL XDM RELOAD
< OK: XDM server configuration reloading
```

## Controlling Access to the XDM Server

Access to the MultiNet XDM server by X11R4 (and later) servers is controlled through the `MULTINET:XDM.ACCESS` file. This file contains the following information:

- Displays the XDM server will manage
- Displays the XDM server will not manage
- Displays and the XDM servers that will manage them
- Macro definitions

The following sections explain how to edit `XDM.ACCESS` to handle direct, broadcast, and indirect requests.

## Handling Direct and Broadcast Requests

To manage X servers that make direct or broadcast requests, add their names to the `XDM.ACCESS` file as follows:

- To manage a specific X server, include a line with the server host name.
- To reject a specific X server, include a line with the server host name preceded by an exclamation point (!).
- To manage any X server that requests management, include a line consisting of an asterisk (\*).

For convenience, you can define lists of hosts in macros in `XDM.ACCESS`. To define a macro, include a line at the top of the file in the following format:

```
%macroname host_list
```

You can then use the macro name (including the leading percent sign (%)) in the file.

The following sample `XDM.ACCESS` file allows any host to be managed by the XDM server.

```
# Access control file for XDMCP connections
#
* #any host can get a login window
```

## Managing X11R3 Displays

Although X11R3 X servers do not allow users to select the host they are going to log into, they can still be managed by the MultiNet XDM server. Once an X server is managed by one host, however, the user has limited ability to change to another host.

To configure the MultiNet XDM server to manage a specific X11R3 display:

1. Add the X11R3 display to the MultiNet XDM server's `XDM.SERVERS` file (see the *Specifying X11R3 Displays* section).
2. Configure the X11R3 display to grant access to the MultiNet host (see the *Setting Up Host Access on the Display* section).
3. Make sure the X11R3 display is not already managed by another XDM server (see the *Ensuring No Other Host Is Managing the Display* section).
4. Reload the XDM server configuration (see the *Reloading the XDM.SERVERS File* section).

## Specifying X11R3 Displays

The XDM server obtains a list of X displays to manage from the `MULTINET:XDM.SERVERS` file whenever it starts or you reload its configuration files. The XDM server obtains its list of X11R3 servers from the file specified in the `DisplayManager.servers` resource in the XDM master configuration file, `MULTINET:XDM.CONFIGURATION`.

Add an entry to `XDM.SERVERS` in the following format:

```
server_name foreign
```

- `server_name` is the name of the X11R3 server you want to manage, in the standard X format (`hostname:server number[.screen number]`).
- "foreign" indicates that `server_name` does not use XDMCP.

The following sample `XDM.SERVERS` file contains entries for two X11R3 servers, `BANZAI:0` and `MIGUEL:0`.

```
banzai.example.com:0 foreign  
miguel.example.com:0 foreign
```

You can also specify classes of displays that share similar requirements.

## Setting Up Host Access on the Display

Before the XDM server can produce a login dialog box on an X11R3 display, it must be granted access to the display via the X host access mechanism.

**Note:** The MultiNet XDM server does not support the MIT-MAGIC-COOKIE-1 mechanism.

The XDM server can only present the `LOGINOUT` dialog on X11R3 servers on which host access restrictions are entirely disabled, or enabled specifically for your MultiNet XDM server host.

For many X servers, you can disable access restrictions with the `xhost` client, or by modifying configuration files before starting the server. Refer to your X server administration documentation for information on granting access to remote hosts.

## Ensuring No Other Host Is Managing the Display

If an X11R3 display is already being managed by another XDM server, either remove the display from the other XDM server's configuration (usually the `XSERVERS` file) or change the X server's host access list so the other XDM server cannot access the display. Cancel the login dialog and verify that the other XDM server's login dialog does not return.

## Reloading the XDM.SERVERS File

Reload the `MULTINET:XDM.SERVERS` file (or stop and restart the XDM server) after you have:

- Specified which X11R3 servers you want to manage in the `MULTINET:XDM.SERVERS` file
- Granted access to your XDM server
- Made sure that no other XDM server is managing the X servers

The XDM server reads the `XDM.SERVERS` configuration file when it starts (see the *Starting the XDM Server* section), restarts (see the *Restarting the XDM Server* section), and when you reload its configuration (see the *Reloading the XDM Configuration* section).

# 20. Configuring MultiNet SNMP Services

This chapter explains how to configure MultiNet SNMP (Simple Network Management Protocol) agents. SNMP agents are the hosts managed by SNMP.

## Understanding SNMP

SNMP is a protocol from which hosts called *network management stations* can obtain and modify information on other network hosts called SNMP agents.

Typical SNMP-based network management systems employ a host that has been configured as an SNMP agent from which the current network configuration of other network nodes can be analyzed and modified. Such management systems often provide graphical interfaces for these tasks.

## SNMP Managers, Agents, and Traps

SNMP-managed networks typically include the following network entities:

<b>Network Management Station (NMS)</b>	Is a node that requests information about, or makes changes to, remote nodes. The NMS interface is completely vendor-specific. MultiNet performs basic SNMP via the <code>MULTINET SET</code> and <code>MULTINET SHOW</code> commands using the <code>/SNMP_HOST</code> qualifiers (see the <i>Performing SNMP Functions with MultiNet</i> section).
<b>SNMP agent</b>	<p>This software allows the local network configuration to be examined or modified by an NMS. MultiNet provides an SNMP agent in the form of the SNMP service.</p> <p>The data managed by an SNMP agent is called the Management Information Base (MIB). The MultiNet SNMP agent supports the current Internet standard MIB known as MIB-II, described in the file RFC1213-MIB.</p>

	<p>MultiNet manages MIBs according to the IETF draft “draft_ref_dhcp_server_mib-02.txt”. Both RFC1213-MIB and “draft_ref_...” are found on the IETF website.</p> <p>MultiNet SNMP agent software is extensible. To have private MIBS served by the MultiNet SNMP agent, develop a shareable image that exports the APIs in the private MIBs plus the routine need to access the MIB variables. See the <i>MultiNet for OpenVMS Programmer’s Reference - SNMP Extensible Agent API Routines</i>.</p>
<p><b>Traps</b></p>	<p>Traps receive requests such as non-authenticated SNMP requests that are not handled directly by the SNMP agent. Traps are sent only to clients configured to receive traps, as defined in the SNMP agent configuration file (SNMPD.CONF). The agent supports all traps defined in the SNMP protocol, except EGP-Neighbor-Loss, Warm-Start, and Enterprise-Specific.</p> <p>Before NMS can obtain or modify network configuration data, the NMS must be authenticated by the agent. The agent compares a community string sent by the manager to the local read or write community strings.</p> <p>The agent must authenticate community strings sent by the NMS before authenticating requests:</p> <ul style="list-style-type: none"> <li>• For read operations, the NMS string must match the agent’s read community string.</li> <li>• For write operations, the NMS string must match the agent’s write community string.</li> </ul>

# Configuring MultiNet SNMP Services

The following is an overview of how to configure a host as an SNMP agent. Each step is discussed in detail following this overview.

1. Enable the SNMP service (see the *Enabling the SNMP Service* section).

2. Configure an SNMP subagent by setting the subagent image (see the *Configuring SNMP Subagents (except AgentX)* section).
3. Edit the SNMP configuration file (see the *Configuration File* section).
4. Restart the master server.

When the MultiNet SNMP agent starts, it obtains configuration data from the `MULTINET:SNMPD.CONF` file. Since the `SNMPD.CONF` file does not exist, you need to edit it using a text editor.

## Enabling the SNMP Service

To enable the SNMP service using the `SERVER-CONFIG` utility:

1. Start `SERVER-CONFIG`:

```
$ MULTINET CONFIGURE /SERVERS
```

2. Enable the SNMP service:

```
SERVER-CONFIG>ENABLE SNMP
```

3. If desired, enable SNMP service on specific VMScluster nodes, or restrict access to the service as described in Chapter 12.

4. Save the modified service configuration.

```
SERVER-CONFIG>SAVE  
[Writing configuration to  
MULTINET_COMMON_ROOT: [MULTINET]SERVICES.MASTER_SERVER]
```

5. Restart the `MULTINET-SERVER` process:

```
SERVER-CONFIG>RESTART
```

6. Exit the utility:

```
SERVER-CONFIG>RESTART
```

## Private MIB Application Program Interface

In addition to SMUX and AgentX, MultiNet's SNMP agent supports subagents serving private MIBs through an application programming interface (API). Under this scheme, anyone willing to have their

private MIBs served by MultiNet's SNMP agent should develop a shareable image that exports the APIs in them in addition to the routines they may need for accessing the MIB variables.

The SNMP API routines are described in Chapter 5 of the *MultiNet for OpenVMS Programmer's Reference, SNMP Extensible Agent API Routines*.

## Configuring SNMP Subagents (except AgentX)

To configure an SNMP subagent on your host using the `SERVER-CONFIG` utility:

1. Start `SERVER-CONFIG`:

```
$ MULTINET CONFIGURE /SERVERS
```

2. Select the SNMP service:

```
SERVER-CONFIG> SELECT SNMP
```

3. Set the subagent-image:

```
SERVER-CONFIG> SET SUBAGENT-IMAGE
```

You can now delete old entries or add new ones. Enter the name of one subagent per prompt, until finished. When finished, press Return at the prompt. *Do not enter the .EXE extension.*

4. Save the modified server configuration and exit.

```
SERVER-CONFIG> SAVE  
[Writing configuration to  
MULTINET_COMMON_ROOT: [MULTINET] SERVICES.MASTER_SERVER
```

5. Restart the `MULTINET-SERVER` process:

```
SERVER-CONFIG> RESTART
```

6. Exit the utility:

```
SERVER-CONFIG> EXIT
```

## SNMP Multiplexing Peers

The SNMP Multiplexing (SMUX) protocol is an SNMP subagent extension protocol. Each subagent or peer registers a MIB subtree with the SNMP Agent. Requests for objects residing in a registered MIB



subtree are passed from the SNMP Agent using the SMUX protocol to the subagent. The subagent passes the results of an SNMP query back to the SNMP Agent. The practical limit to the number of peers is 30.

Enabling SMUX (`DEFINE/SYSTEM/EXEC MULTINET_SNMP_SMUX 1`) when there are no SMUX subagents to use it can interfere with walking of the SNMP management base due to the SMUX MIB returning `NoSuchName` when no subagents exist. SMUX is an historical protocol, and should not be enabled unless there are subagents that will be using it. Specific items in the SNMP management base that appear after the SMUX MIB can still be queried when they are accessed from the start of their management base.

The SNMP server only accepts SMUX connections from peers listed by IP address in the `SNMPD.CONF` file. To enable SMUX support, issue the following command before starting SNMP:

```
$ DEFINE/SYSTEM/EXECUTIVE MULTINET_SNMP_SMUX 1
```

## SMUX\_PEER ip-address

The SNMP agent listens on TCP port 199 for peer connections, while the connection to the SNMP client is over UDP port 161, with traps sent over UDP port 162. Multiple peers registering the same subtree are each assigned a priority, and the agent can send multiple variables in a single request. The SMUX protocol is described in RFC 1227.

# SNMP Agent Extensibility (AgentX) Peers

The SNMP agent listens on TCP port 705 for subagent connections. The AgentX framework consists of a single processing entity called the master agent. This master agent, available on the standard transport address, sends and receives SNMP protocol messages in an agent role but has little or no direct access to management information. While some of the AgentX protocol messages appear similar in syntax and semantics to the SNMP, remember that AgentX is not SNMP. Refer to RFCs 2741 and 2742 for complete AgentX information. The SNMP server only accepts AgentX connections from peers listed in the `SNMPD.CONF` file. To enable AgentX support, issue the following command before starting SNMP:

```
$ DEFINE/SYSTEM/EXECUTIVE MULTINET_SNMP_AGENTX 1
```

or

```
$ MULTINET CONFIGURE/NET
NET-CONFIG>SET SNMP-AGENTX TRUE
```

## Setting Up MultiNet to Use HP Insight Manager

HP Insight Manager support has been added to MultiNet. The HP Insight Manager (CIM) uses the SNMP extensibility provided by Agent X to allow remote examination and notification of system conditions that may need attention. Remote management agents like CIM allow systems administration personnel to manage more systems while still meeting response time goals by providing access to critical information from a central location. The remote management agent communicates with the SNMP agent on the system being managed, which then sends the request to a program specifically designed to manage a particular component of the system.

Customers desiring to run HP Insight Manager (OpenVMS 7.1 and later) need to obtain TCP/IP Services v5.1 or greater for OpenVMS from HP in order to get the TCPIP\$ESNMP\_SHR.EXE, and TCPIP\$HR\_MIB.EXE images. Contact HP Enterprise ([www.hpe.com](http://www.hpe.com)) to get the kit for TCP/IP Services v5.1 or greater.

HP Insight Manager is only available on Alpha and Itanium systems with VMS 7.1 or higher. Please follow these instructions.

1. Install MultiNet
2. Install HP Management Agents
3. Copy the TCP/IP Services kit from HP's site, checking for latest patches
4. Extract TCPIP\$ACCESS\_SHR.EXE and TCPIP\$ESNMP\_SHR.EXE (PRODUCT EXTRACT FILE/SELECT=filename) from the TCPIP Services kit and place them in SYS\$SHARE:.
5. Copy TCPIP\$ESNMP\_SHR.EXE to SYS\$SHARE:UCX\$ESNMP\_SHR.EXE
6. Extract TCPIP\$HR\_MIB.EXE from the TCPIP Services kit and place in SYS\$SYSTEM:

```
$ DEFINE/SYSTEM/EXECUTIVE MULTINET SNMP_AGENTX 1
$ DEFINE/SYSTEM TCPIP$AGENTX_INET_PORT 705
$ DEFINE/SYSTEM TCPIP$AGENTX_LOCAL_PORT 705
```

**Note:** Make sure all files extracted from the TCP/IP Services kit have WORLD:RE protection.

7. Add the following to `MULTINET:SNMPD.CONF`:

```
AGENTX_PEER 127.0.0.1
community elmginkgo 127.0.0.1 read (and other community strings as needed)
```

8. Comment out `SMUX_PEER` from `MULTINET:SNMPD.CONF`

9. Restart SNMP with this command:

```
$ MULTINET NETCONTROL SNMP RELOAD
```

or

```
$ START MULTINET
```

10. Start HP Insight Manager

11. Run `/process=HR_MIB SYS$SYSTEM:TCPIP$HR_MIB`

The Host Resources MIB (RFC 1514) supplied with TCP/IP Services will now work with SNMP.

The new ESNMP client interface in HP's TCP/IP services v5.1 or greater uses Agent X to allow others to provide additional objects for SNMP to manage. HP's Insight Management Agents for OpenVMS are written to use the ESNMP client interface, hence the addition of Agent X protocol allows them to be used with MultiNet.

By using the ESNMP library, or Agent X directly, writers of TCP/IP services can allow the state of the service to be queried and controlled remotely. This can be useful if the service does not have a user interface, or runs under batch, or as a detached process.

## Configuration File

The SNMP configuration file `SNMPD.CONF` is located in the `MULTINET_ROOT` directory. The SNMP file defines:

- Values for a subset of MIB management objects
- Clients and communities who can access the SNMP agent
- MIB access privileges for each client and community
- Authentication Failure, Link Up, and Link Down traps' status
- Originating addresses for traps
- SMUX peer details
- Agent X peer details

After editing the configuration to fit your needs, stop and restart the SNMP agent so that the changes can take effect. If you do not edit the configuration file, the SNMP agent uses default values.

## File Format

Follow these guidelines when entering data in the SNMP configuration file:

- Allow one line for each item.
- Enter information in any order, in upper- or lowercase.
- Enter variable string information (`id-string` and `contact-name`) in upper- or lowercase, depending on the operating system. Some SNMP clients on your network (such as those running UNIX) might require information in a specific case.
- Use a pound sign (#) or an exclamation point (!) to denote comments. SNMP ignores all information following these characters.
- Place quotation marks (" ") around strings that contain spaces or that require more than one line in the file, and around the comment characters when used as regular characters.

## Values for MIB Objects

To define the values of several MIB objects in the SNMP configuration file, use the corresponding keywords listed in the below table.

MIB object name...	Has keyword...
<code>system.sysDescr</code>	SYSDESCR
<code>system.sysContact</code>	SYSCONTACT
<code>system.sysLocation</code>	SYSLOCATION
<code>if.ifTable.ifEntry.ifDescr</code> and <code>if.ifTable.ifEntry.ifSpeed</code>	INTERFACE
<code>system.sysServices</code>	SYSSERVICES

The following paragraphs explain how to define each item.

**SYSDESCR** [ *id-string* ]

The *id-string* is the full name of the hardware, operating system, and networking software. For example:

```
SYSDESCR "AlphaServer 8400, VMS V7.1, Process Software MultiNet for OpenVMS"
```

If you omit the *id-string*, MultiNet tries to obtain this information from your current system. If the attempt fails, the default is System description is unknown. Try again, entering a different *id-string*.

#### **SYSCONTACT [ *contact-name* ]**

The *contact-name* specifies the person to contact for the host, and how you can contact this person (such as by mailbox address). For example:

```
SYSCONTACT "John Smith, X 1234, smith@process.com"
```

The default is System contact is unknown at this time. Try again, entering a different *contact-name*.

#### **SYSLOCATION [ *system-location* ]**

The *system-location* specifies the geographical location of the host. For example:

```
SYSLOCATION "Main Street, Anytown, MA"
```

The default is: System location is unknown at this time. Try again, entering a different *system-location*.

#### **INTERFACE [ *line-id line-speed description* ]**

The *line-id* specifies the line identification for the IP layer network device. The *line-speed* specifies the line speed in bits per second. The *description* is the manufacturer's name, product name, and hardware version for the interface. For example:

```
INTERFACE SE0-1 10000000 "DELQA Ethernet Controller Version 1.0"
```

#### **SYSSERVICES [ *services-set-number* ]**

The *services-set-number* default is 72. RFC 1213, Management Information Base for Network Management of TCP/IP-based Internets: MIB-II, explains how to calculate the value of *services-set-number*.

#### **HOSTID *ip-address***

Specifies the IP address to use as the source address for traps sent either from the SNMP Agent or from the TRAP\_GEN program. If this is not specified the address of the first interface (often SE0) is used. When this is specified it is checked against the addresses of the interfaces present on the system.

## Community Parameters

The SNMP configuration file must contain the following information for each client permitted access to the SNMP agent:

**COMMUNITY *community-name internet-address type***

community-name	Specifies the name of the community to which the client belongs.
internet-address	Specifies the client's internet address.  If you enter 0.0.0.0, any address can use the community. The internet address can be optionally followed by /mask for READ and WRITE.
type	Defines the access profile as one of the following: <ul style="list-style-type: none"><li>• READ - The client can retrieve data from the MIB on this host.</li><li>• WRITE - The client can retrieve data from and write data to the MIB on this host.</li><li>• TRAPS - The client will receive all enabled traps.</li></ul>

The following is an example of some community parameters defined in the configuration file.

```
community northeast 192.168.4.56 READ
community northeast 192.168.220.1 WRITE
community southwest 192.168.23.1 WRITE
community southwest 192.168.23.1 TRAPS
```

- Client 192.168.4.56 in the northeast community has READ access to the MIB, while client 192.168.220.1 in the same community has WRITE access.
- Client 192.168.23.1 belongs to the southwest community. This community has WRITE access to the MIB and can receive all traps.

# Template Configuration File

SNMP Services provides a `TEMPLATE_SNMPD.CONF` file in `MULTINET_COMMON`: `[MULTINET]` that you can use as a basis:

```
! SNMP Agent (SNMPD) Configuration File (template)
!
! System description: sysdescr <id string>
! Typically the id string would include:
! VAX cpu model (such as MicroVAX II, VAX 8650, etc)
! VMS and version number
! "Process Software MultiNet for OpenVMS Version 5.5"
!
sysdescr "place system description string here"
!
! System Contact: syscontact <contact name>
!
syscontact "place name, phone number, and mail address of administrator
here"
!
! System Location: syslocation <location>
!
syslocation "place system location information here"

! Line Interfaces Information: interface <line-id> <line speed>
! <description>
! Note: You usually need not define these. SNMPD provides good defaults.
!
! <line-id> is one of LPB-, ETHER-, UNA-, QNA-, BNA-, SVA-, MNA-,ISA-,KFE,
! MXE-, ERA-, EWA-,CEC-, EIA-, CLIP-, ELA-,FDDI-,MFA-,FZA-, FAA-, FEA-,
! FQA-, FPA-, TR-, TRA-,TRE-, TRW-,PRO-, HYP-, DSV-, DSB-,DST-, X25-,SLIP,
! DECnet-, PPP-,PSD- followed by a unit number. Note that the unit number
! may be an encoding of the controller when the device is an ethernet
! adapter.
!
! (A = 0, B= 1, C=2, etc.)
!
! <line-speed> is an integer in bits per second of the data rate of the
! device
!
! <description> is a quoted string describing the device.
!
!interface una-0 10000000 "HEWLETT-PACKARD DELUA Ethernet controller"
!
! Communities:
! community <community name> <internet address>
! <READ | WRITE | TRAPS>
!
community readers 192.168.1.2 READ
community netman 192.168.2.3 WRITE
community nettraps 192.168.3.4 TRAPS
```

```

!
! To disable authentication traps, remove the "!" from the following
! line.
!no-auth-traps
!
! To disable link status traps, remove the "!" from the following
!line.
!no-link-traps
!
! SMUX Peers:
! AGENTX_PEER <ip-address>
! SMUX_PEER <ip-address>
!
AGENTX_PEER 192.168.6.7
SMUX_PEER 192.168.4.5
SMUX_PEER 192.168.5.6

```

`AGENTX_PEER ip-address` - The SNMP server only accepts AGENT X connections from peers listed by IP address in the `SNMPD.CONF` file. Use the following syntax in the file:

```
AGENTX_PEER ip-address
```

The `COMMUNITY`, `SMUX_PEER`, and `AGENTX_PEER` statements in the `SNMPD.CONF` file can take an optional mask after the internet address. The mask should be separated from the internet address with a / (slash). Valid values are from 0 to 32, with 32 being the default. Although the `TRAPS` community accepts a mask, it is not used currently.

For example:

```
community ournet 192.168.1.10 write !implied /32
community ourmgr 192.168.1.0/24 read
```

The `/24` specifies that only the first 24 bits must match. In this example, IP addresses from 192.168.1.0 to 192.168.1.255 can use the `community ourmgr`. The mask should be separated from the internet address with a / (slash). Valid values are from 0 to 32, with 32 being the default. A more restrictive IP address may be used within a less restrictive one. For example:

```
community process 192.168.6.0/24 READ
community process 192.168.6.42 WRITE
```

This allows all nodes in 192.168.6 to have `READ` access with the community name `process` and 192.168.6.42 to have `WRITE` access with the community name `process`.

The following command can be used to display enabled SNMP Agent X subagents in the output of the `SHOW` command:

```

$ MULTINET CONFIGURE /NETWORK
$ SET SNMP-AGENTX TRUE
$ SET SNMP-AGENTX FALSE

```



TRUE enables SNMP Agent X service; FALSE disables SNMP Agent X service. A line displays in the output of the SHOW command if SNMP Agent X subagents are enabled.

## Sending SNMP Traps from MultiNet

SNMP traps can be sent from MultiNet in the following manner:

Define the symbol:

```
$ trap_gen := $multinet:trap_gen
```

Then type:

```
$ trap_gen enterprise generic-trap specific-trap [trap-specific-values...]
```

*enterprise* identifies the location in the MIB tree that this trap pertains to. An example would be 1.3.6.1.4.105.3, which denotes a location in Process Software's portion of the MIB tree.

*generic-trap* is an integer representing the generic trap value.

*specific-trap* is an integer representing the specific trap value.

*trap-specific-values* are arbitrary strings separated by spaces that are passed to the agent receiving the trap as octet strings.

The TRAP\_GEN program uses the trap community definitions in the MULTINET:SNMPD.CONF file to determine where to send the trap.

There is also a program available that will listen for traps and format them for display. In order to invoke this program, run MULTINET:TRAP\_LISTEN. It prompts for an optional file to log information to (default is the terminal) and the port number to listen on (default is 162).

## Disabling Traps

All traps that the SNMP agent supports are initially enabled. You can disable traps by editing the configuration file. These changes take effect the next time you start the agent. This table shows how to disable traps.

Disable this trap...	By entering...
Authentication Failure	no-auth-traps
Link Up	no-link-traps
Link Down	no-link-traps

**Note:** SNMP clients can enable or disable the Authentication Failure trap while the SNMP agent is running. These clients must have WRITE community access.

## Generating Traps

To generate an SNMP trap, define the symbol:

```
$ TRAP_GEN := $MULTINET:TRAP_GEN
```

Then type:

```
$ TRAP_GEN ENTERPRISE GENERIC-TRAP SPECIFIC-TRAP [TRAP-SPECIFIC-VALUES...]
```

<code>enterprise</code>	Identifies the location in the MIB tree that this trap pertains to. An example would be: 1.3.6.1.4.105.3, denoting a location in Process Software's portion of the MIB tree.
<code>generic-trap</code>	Is an integer representing the generic trap value.
<code>specific-trap</code>	Is an integer representing the specific trap value.
<code>trap-specific-values</code>	Are arbitrary strings separated by spaces that are passed to the agent receiving the trap as octet strings.

The TRAP\_GEN program uses the trap community definitions in the MULTINET : SNMPD . CONF file to determine where to send the trap.

To specify a particular ip-address for SNMP traps to originate from put the following line in MULTINET : SNMPD . CONF

```
HOSTID ip_address
```

The ip\_address specified is checked against those on the system when the line is parsed.

## SNMP Log File

When the SNMP agent starts up, it creates a log file called MULTINET : SNMPSERVER . LOG. This file contains information about the activities of the SNMP agent, such as:

- The time the agent starts up and shuts down.
- When SMUX peers open or close a connection, and register or de-register a MIB tree.
- Any errors found in the SNMP configuration file.
- Any errors that occur when the agent is running.

## Start, Shutdown, or Reload the SNMP Configuration without Rebooting

To start, shutdown, or reload the SNMP configuration using NETCONTROL:

```
$ MULTINET NETCONTROL SNMP START
$ MULTINET NETCONTROL SNMP SHUTDOWN
$ MULTINET NETCONTROL SNMP RELOAD
```

MultiNet has a separate SNMP\_AGENT process. Once the SNMP service is enabled via the MULTINET CONFIGURE / SERVERS command, the SNMP\_AGENT process can be accessible via the NETCONTROL commands. You can also enable the SNMP service using the MULTINET CONFIGURE / MENU command.

# Performing SNMP Functions with MultiNet

The `MULTINET SET` and `MULTINET SHOW` commands accept the `/SNMP_HOST` qualifier for using remote host information.

The below table shows qualifiers you can use with the `/SNMP_HOST` qualifier in `MULTINET SHOW` commands to obtain information from remote SNMP agents.

Qualifier	Description
<code>/COMMUNITY_NAME="string"</code>	<p>Specifies the community name string sent with this command to the remote host. The default is public.</p> <p><b>Note:</b> The case of the community name must match what is specified in <code>SNMPD.CONF</code>. The name must be specified in quotes unless it is all uppercase.</p>
<code>/CONNECTIONS [=ALL]</code>	<p>Displays network connections. If you use the <code>=ALL</code> argument, <code>MULTINET SHOW</code> also displays sockets on which servers are listening.</p>
<code>/ARP</code>	<p>Displays the Ethernet Address Resolution Protocol tables. This qualifier (or <code>/ROUTE</code>) must precede all other qualifiers.</p>
<code>/MIB_VAR [=variable_name]</code>	<p>Displays the value of the SNMP MIB variable, <code>variable_name</code>. The value can be any MIB-II variable described in RFC-1213. If you omit <code>variable_name</code>, all MIB variables are displayed.</p>
<code>/ROUTE</code>	<p>Displays routing information for the IP protocol. This qualifier (or <code>/ARP</code>) must precede all other qualifiers.</p>

`/STATISTICS[=protocol]`

Causes MULTINET SHOW to display either network interface statistics or protocol statistics or both, as defined in MIB-II. If you specify /STATISTICS without a value, INTERFACE statistics are displayed.

# 21. Configuring Kerberos Authentication Service (DEPRECATED)

**CAUTION!** This chapter used to document MultiNet's Kerberos V4 server, which has been deprecated and is no longer available. Please see Chapter 3, *Using Kerberos Authentication*, in the MultiNet User's Guide for information on a more modern implementation.

# 22. X11 Gateway Configuration

The MultiNet X11-Gateway program provides X (X Window System) connectivity between a DECnet-only host and an IP-only host by a MultiNet node as an application gateway. The X11-Gateway is bidirectional; it functions as a gateway from a DECnet-only X client to an IP-only X server, or vice versa.

The gateway node requires MultiNet and DECnet software only. There is no requirement for the gateway to be running any X software. The gateway software can support multiple X Windows connections simultaneously.

## Concepts

Before configuring the X11-Gateway, be sure you understand the following terms:

<b>Client</b>	<b>The node executing the X application</b>
<b>Gateway</b>	The node connected to an IP network and a DECnet network. Information from the <i>client</i> on one network is passed through the <i>gateway</i> to the <i>server</i> on the other network.
<b>Server</b>	The node running the X server software. (Typically a host with a mouse, keyboard, and at least one bit-mapped screen.)
<b>Server number</b>	A unique number identifying the X11-Gateway to MultiNet, DECnet, and the X software on the client and server nodes. Each configured gateway is assigned a unique server number by the system manager.

To avoid conflicts with current and future versions of DECwindows software, use numbers beginning with 10 and increment for each new gateway. For example, assign the number 10 to the first X11-Gateway, the number 11 to the second, and so on. As you remove X11-Gateways from the system, you can reuse their server numbers.

# Allowing an IP Client Access to a DECnet Server

To configure the X11-Gateway host to allow an IP client access to a DECnet server:

1. Choose an X11-Gateway server number between 10 and 999.
2. Create a TCP port number by adding 6000 to the server number. For example, if the server number is 13, the TCP port number is 6013. TCP port 6000 is used by DECwindows servers. Select port numbers starting at 6010 to avoid conflicts with DECwindows.
3. Add the X11-Gateway service to the list of TCP/IP services using the Server Configuration Utility (SERVER-CONFIG). Information about using this utility and its commands is provided in Chapter 12.

The prefix X11-GATEWAYxxx is added to the name of the service you install; xxx is the server number you selected in Step 1. For example, for server number 10, the service you add using SERVER-CONFIG is X11-GATEWAY10. SERVER-CONFIG provides the X11-GATEWAY13 service, but the number 13 has no significance; this service is provided only as an example. You can use this service or any number of your choice between 10 and 999. If you chose 13 as the server number in Step 1, you can enable the existing X11-GATEWAY13 service using SERVER-CONFIG. There is no need to add this service, as it has already been added it. The following example adds an X11-Gateway server number of 12.

```
$ MULTINET CONFIGURE/SERVER
MultiNet Server Configuration Utility 5.6
[Reading in configuration from MULTINET:SERVICES.MASTER_SERVER]
SERVER-CONFIG>ADD X11-GATEWAY12
[Adding new configuration entry for service "X11-GATEWAY12"]
Protocol: [TCP] RETURN
TCP Port number: 6012
Program to run: MULTINET:X11-GATEWAY.EXE
[Added service X11-GATEWAY12 to configuration]
[Selected service is now X11-GATEWAY12]
SERVER-CONFIG>RESTART
Configuration modified, do you want to save it first ? [YES] RETURN
[Writing configuration to
MULTINET_COMMON_ROOT:[MULTINET]SERVICES.MASTER_SERVER]
%RUN-S-PROC_ID, identification of created process is 20E0026B
SERVER-CONFIG>EXIT
```

4. Define the X11-Gateway logical names. The X11-Gateway accepts connections from the IP network and routes the X protocol requests to a specific DECnet X server. Specify the server using the following logical names:

```
MULTINET_XGATEWAY_TCPIP_server_number_HOSTNAME
```



Specifies the DECnet host name. This logical name must be defined, but do not include the colons from the DECnet host name.

#### **MULTINET\_XGATEWAY\_TCPIP\_server\_number\_SERVER**

The X server number of the node where the X client application is displayed. Most hosts run one X server, which is designated as server 0 (zero). This logical name is optional if the DECnet X server number is zero. The X11-Gateway assumes a DECnet X server number of zero if you do not define this logical. You should define the logical if the DECnet X server number is not zero.

In the following example, the X11-Gateway server number is 12. The X11-Gateway accepts connections from the IP network and gateways them to X server 1 on the DECnet node BRONX. Assign values to these logical names using commands like the following examples:

```
$ DEFINE/SYSTEM/EXEC MULTINET_XGATEWAY_TCPIP_12_HOSTNAME BRONX
$ DEFINE/SYSTEM/EXEC MULTINET_XGATEWAY_TCPIP_12_SERVER 1
```

Insert these logical name definitions in the system startup procedure so they are invoked after the DECnet and MultiNet startup procedures execute.

## Running an IP Client on a DECnet Server

To bring up the X application on the IP-client-to-DECnet-server configuration:

1. On the DECnet server, authorize DECnet connections from user SYSTEM on the gateway node. If the MULTINET\_SERVER process on the gateway node has been started under a user name other than SYSTEM, that user should also be authorized. A less secure, but more reliable, method is to authorize the "\*" user. See the *X11-Gateway Security* section.

On UNIX systems, use the `xhost` command to provide connection authorization. On ULTRIX or OpenVMS, use the Session Manager.

2. On the IP client, set the display variable to point to the X11-Gateway host. Use the X11-Gateway server number for the display server number.

On UNIX systems, use the `setenv` command to modify the DISPLAY environment. On OpenVMS systems, use the `SET DISPLAY` command.

3. On the IP client, start the X application.

For example:

- The IP X client node is the UNIX node `pelham.example.com`. The X11-Gateway node is `metro.example.com` (TCP/IP) and `METRO::` (DECnet). The X11-Gateway server number is 12. `BRONX::` is an OpenVMS DECnet X server.
- On the `BRONX::` node, the user authorizes protocol DECNET, node `METRO`, and user "\*" using the Security pull-down menu in the Session Manager.
- On the `pelham.example.com` UNIX node, the user runs `setenv` to set the `DISPLAY` environment variable to the value `amtrak.flowers.com:12.1`. The user can then invoke an X application.
- The X application appears on the `BRONX::` node.

## Allowing a DECnet Client Access to an IP Server

To configure the X11-Gateway to allow a DECnet client access to an IP server:

1. Choose an X11-Gateway server number as described in *Allowing an IP Client to Access a DECnet Server*.
2. Add the X11-Gateway to the list of DECnet objects. The object name for the X11-Gateway has the value `X$Xserver_number`. For example, for a server number of 17, set the object with this command:

```
$ RUN SYS$SYSTEM:NCP
NCP> DEFINE OBJECT X$X17 NUMBER 0 FILE MULTINET:X11-GATEWAY.EXE
NCP> SET OBJECT X$X17 NUMBER 0 FILE MULTINET:X11-GATEWAY.EXE
```

If the DECnet default account is disabled, the command should include a valid `USERNAME` and `PASSWORD` on the gateway system. In this example the X11-Gateway server number is 17:

```
$ RUN SYS$SYSTEM:NCP
NCP> DEFINE OBJECT X$X17 NUMBER 0 FILE -
MULTINET:X11-GATEWAY.EXE USER SYSTEM PASS systempassword
NCP> SET OBJECT X$X17 NUMBER 0 FILE MULTINET:X11-GATEWAY.EXE -
USER SYSTEM PASS systempassword
```

3. Define the logical names. The X11-Gateway accepts connections from the DECnet network and directs X protocol requests to a specific IP X server. Specify the server is using the following logical names:

**MULTINET\_XGATEWAY\_DECNET\_server\_number\_HOSTNAME**

Specifies the IP X server host name. You must define this logical name.

### **MULTINET\_XGATEWAY\_DECNET\_server\_number\_SERVER**

Specifies the X server number, which is typically set at 0 (zero) to indicate that a single server is being used. If a second server is in use, set this value to 1, and so on. If this logical name is not defined, the default value is 0. For example, the X11-Gateway server number is 17. The X11-Gateway accepts connections from the DECnet network and directs them to X server number 1 on the IP node englewood-nj.example.com. The logical names are then defined as:

```
$ DEFINE/SYSTEM/EXEC MULTINET_XGATEWAY_DECNET_17_HOSTNAME -  
_ $ ENGLEWOOD-NJ.EXAMPLE.COM  
$ DEFINE/SYSTEM/EXEC MULTINET_XGATEWAY_DECNET_17_SERVER 1
```

Insert these logical name definitions into the system startup procedure so the definitions occur after the invocation of the DECnet and MultiNet startup procedures.

## **Running the DECnet Client on the IP Server**

To bring up the X application on the DECnet-client-to-IP-server configuration:

1. On the IP server, authorize IP connections from the gateway node. X-over-IP does not provide user name information. If a user name is required as part of the authorization (for example, on OpenVMS) use a "\*" value. Connection authorization is usually accomplished with the `xhost` command on UNIX systems, or with the OpenVMS or ULTRIX Session Manager.
2. On the DECnet client, set the display variable to point to the X11-Gateway host. The X11-Gateway server number should be used for the display server number. On UNIX hosts (including ULTRIX) use the `setenv` command; on OpenVMS systems, use the `SET DISPLAY` command.
3. On the DECnet client, execute the X Windows application.

For example:

- The DECnet X client is an OpenVMS node `METRO::`. The X11-Gateway node is `DENISE::` (DECnet) and `DENISE.EXAMPLE.COM` (TCP/IP). The X11-Gateway server number is 17. The IP X server is the UNIX host `englewood-nj.example.com`.
- On `englewood-nj` the user authorizes node `denise.example.com` by entering the command:

```
% xhost +denise.example.com
```

- On `METRO`, the user issues the command:

```
$ SET DISPLAY/CREATE/NODE=DENISE/TRANS=DECNET/SERVER=17
```

- The user can then invoke an X application, which appears on the `englewood-nj` server.

## X11-Gateway Security

The X11-Gateway node does not attempt to restrict connections it receives from the network. As a result, any node or user on the client side of the gateway can access the server, essentially allowing a client user to monitor all activity on the X server via the X11-Gateway.

For IP-client-to-DECnet-server connections, you can reduce risk by using the `ACCEPT-HOSTS/ACCEPT-NETS` capabilities of the MultiNet master server on the gateway host. For more information, see Chapter 12.

For DECnet-client-to-IP-server connections, your risk can be reduced by using the NCP utility to limit access to the gateway host.

Process Software does not recommend running the X11-Gateway on untrusted networks unless other restrictions have been imposed by the system manager.

## X11-Gateway Debugging

The best programs for testing client-to-gateway to server connectivity are based on the Xlib routines (as opposed to widget toolkits).

The `ICO` program is available on most X implementations (for example, `/usr/bin/X11/ico` or `DECW$EXAMPLES:ICO`) and works well for debugging problems. The `ICO` program opens a window and causes an icosahedron to bounce around the window. When this program works, X works as well. Exit the OpenVMS version of this program by pressing `Ctrl/Y` in the window from which you invoked `ICO`. You can also use the MultiNet `X11DEBUG` command to debug OpenVMS IP client problems.

## Selected Error Numbers from ERRNO.H

The table below lists error values from the `ERRNO.H` file.

Error	Value	Description
ENETUNREACH	51	The IP network you are trying to contact is currently unreachable.
ECONNRESET	54	The connection was reset by the remote node. This typically occurs when the remote host has rebooted and the local host attempts to transmit on a stale connection.
ETIMEDOUT	60	The connection timed out during the open.
ECONNREFUSED	61	The connection was refused. This occurs when a connection is attempted to a nonexistent server process.
EHOSTUNREACH	65	There is no route to the host you are trying to contact.

## X11-Gateway Error Messages

The X11-Gateway node transmits NETWORK class operator messages when an error is encountered. You can change the level of information supplied by X11-Gateway messages by defining the logical name MULTINET\_XGATEWAY\_DEBUG\_LEVEL in the system table. Set this value to:

- 0 - to receive fatal errors
- 1 - for debugging messages
- 2 - for informational messages

For example, to select debug level 1:

```
$ DEFINE/SYSTEM/EXEC MULTINET_XGATEWAY_DEBUG_LEVEL 1
$ @MULTINET:START_SERVER
```

If the logical name does not exist, the DEBUG level defaults to a value of zero. All error messages from the X11-Gateway are prefixed with Xgateway:. The errno values can be translated by examining *ERRNO.H Error Values*, or by consulting *ERRNO.H* in the *MultiNet Programmer's Guide*. Status values are OpenVMS error values that you can examine using the command WRITE SYS\$OUTPUT F\$MESSAGE (Status).

# 23. Configuring DECnet-Over-IP Circuits

A special DECnet device driver allows the MultiNet system manager to configure a DECnet line and circuit between two cooperating MultiNet systems across an arbitrary IP network. This special driver encapsulates DECnet packets in UDP datagrams for transport via the IP protocols, in much the same way as VMS PSI encapsulates DECnet packets in X.25 when doing Data Link Mapping.

## Using the Configuration Tools

MultiNet provides `DECNET-CONFIG`, the command-line configuration utility invoked with the `MULTINET CONFIGURE /DECNET` command, for configuring DECnet-over-IP connections:

For details about this utility, see Chapter 2, the `DECNET-CONFIG` chapter of the *MultiNet Administrator's Reference*, and the online command reference (invoked with the `HELP MULTINET` command).

Once configured, a DECnet-over-IP circuit comes up automatically when the hosts on each side of the DECnet connection are rebooted.

When configuring a DECnet-over-IP circuit, you are prompted for:

- The IP address of the host on the opposite side of the connection
- The `COST` that DECnet assigns to the circuit
- The `HELLO TIMER` that DECnet should use on this circuit

For proper DECnet operation, the `COST` and `HELLO TIMER` must be the same on both sides of the connection.

If you configure a DECnet-over-IP link, and you also run DECnet over another Ethernet interface, you must configure your system as a router as follows:

```
$ RUN SYS$SYSTEM:NCP
NCP>DEFINE EXECUTOR TYPE ROUTING IV
```

Non-routing hosts can only communicate with the hosts reachable via the circuit with the lowest cost. If, for example, your Ethernet circuit has a cost of 4, and your DECnet-over-IP link has a cost of 1 (the default), your system will be unable to communicate with hosts accessible over the Ethernet link.

Full DECnet routing functionality is supported only on VAX systems. This is a Hewlett-Packard restriction, not one imposed by MultiNet. For more information, refer to SPD 48.48.04

# Examples of Connecting Two Systems

The following examples show how to make a connection between two systems, betty.example.edu (IP address 192.0.0.6) and wilma.example.com (IP address 128.0.0.125). The first example shows the circuit configuration on the host "betty":

```
$ MULTINET CONFIGURE /DECNET
MultiNet DECNET Circuit Configuration Utility 5.6
[Reading in configuration from MULTINET:DECNET-CIRCUITS.COM]
DECNET-CONFIG>ADD
[Adding new configuration entry for DECnet circuit "TCP-0-0"]
Destination IP Address: [NONE] 128.0.0.125
DECnet circuit cost: [1] 1
DECnet hello timer (in seconds): [300] 300
[TCP-0-0 => 128.0.0.125 (Cost=1, Hello Timer=300)]
DECNET-CONFIG>EXIT
[Writing configuration to MULTINET:DECNET-CIRCUITS.COM]
$
```

The next example shows the circuit configuration on the host "wilma":

```
$ MULTINET CONFIGURE /DECNET
MultiNet DECNET Circuit Configuration Utility 5.6
[Reading in configuration from MULTINET:DECNET-CIRCUITS.COM]
DECNET-CONFIG>ADD
[Adding new configuration entry for DECnet circuit "TCP-0-0"]
Destination IP Address: [NONE] 192.0.0.6
DECnet circuit cost: [1] 1
DECnet hello timer (in seconds): [300] 300
[TCP-0-0 => 192.0.0.6 (Cost=1, Hello Timer=300)]
DECNET-CONFIG>EXIT
[Writing configuration to MULTINET:DECNET-CIRCUITS.COM]
$
```

# DECnet Encapsulation Over Unreliable Networks

Both TCP and DECnet guarantee reliable delivery of data over unreliable networks. This is accomplished through an acknowledgment scheme in which the receiver of the data tells the transmitter of the data that the data has arrived intact. If the acknowledgment is not received within a certain period of time (known as the *retransmission timer*), the data is resent by the transmitter.

The data transmitter must make a good estimate of the retransmission timer. Too long an interval causes unnecessary waits before retransmission occurs, reducing the usable bandwidth of the network. Too short an interval means the transmitter might retransmit data that was merely delayed in transit, unnecessarily loading the network.

- TCP chooses the retransmission timer as a function of the mean and the variance in the *roundtrip time* so that a statistically small percentage of packets are unnecessarily retransmitted.
- DECnet chooses the retransmission timer as the product of the round trip time (with a minimum of one second) and the *delay factor*.

DECnet's scheme (or schema?) does not take into account the variance of the round-trip time and estimated roundtrip times of less than one second.

DECnet uses a very conservative value for the delay factor to avoid any unnecessary retransmissions into congested, low-speed links. A single lost packet results in a delay of at least five seconds in DECnet traffic. Over high-speed, low-latency circuits with any substantial packet loss, this delay results in a severe performance degradation.

If your network has these characteristics, you can substantially increase performance by reducing the delay factor using NCP on each of your nodes. Doing so gives DECnet a more aggressive retransmission strategy which results in shorter delays following a lost packet. Specify the delay factor in units of 1/16th of the mean round-trip time using the NCP EXECUTOR parameter DELAY FACTOR.

Reasonable factors range from 1.5 to 3, or DELAY FACTOR values from 24 to 48. A retransmission factor of 1.5 is very aggressive and about as small as is reasonable before many extra retransmissions occur; a value of 3 more closely mimics TCP's behavior over lines which have typical variances in the roundtrip time.

You can set the DELAY FACTOR to 24/16ths (1.5) using the following NCP commands:

```
$ MCR NCP
NCP>SET EXECUTOR DELAY FACTOR 24
NCP>DEFINE EXECUTOR DELAY FACTOR 24
NCP>EXIT
$
```



# Using MULTINET SET /DECNET

Use the `MULTINET SET/DECNET` command to configure the `TCPAx`: DECnet devices for running DECnet over UDP.

**Note:** You should configure DECnet circuits using `DECNET-CONFIG`, which invokes `MULTINET SET/DECNET` as part of network startup to set up the DECnet link. You can use this utility to change the configuration once the network has started.

# 24. Configuring the MultiNet NFS v2 Server

This chapter describes how to configure and maintain the MultiNet NFS v2 server, the MultiNet software that allows users of OpenVMS computers to export files to a variety of computers.

This chapter refers to the MultiNet NFS server and NFS client software as the *NFS Server* and *NFS Client*, and to the OpenVMS server host as the *server*, and to NFS client hosts as *clients*.

## Understanding the MultiNet NFS Server

The NFS server is a high-performance OpenVMS implementation of Sun Microsystems' Network File System (NFS) protocol. It allows client computers running a variety of operating systems to remotely access files on an OpenVMS server. To users on a client system, all mounting and access operations are transparent, and the mounted directories and files appear to reside on the client.

After the NFS server is installed, the system manager configures the server and its clients to allow network file access. The NFS server includes configuration utilities for this purpose.

The NFS server is exceptionally fast due to parallel request processing, a file and directory cache between the file systems and the network, and an optional writeback cache feature. For example, because the NFS server can process many client requests simultaneously, a single client does not interfere with the requests of others.

## Servers and Clients

An NFS server system is an OpenVMS system that makes its local files available to the network. A client is a host that accesses these files as if they were its own. Typical clients include:

- Systems running UNIX or Linux
- PCs running Microsoft Windows or MacOS
- OpenVMS computers running the MultiNet NFS client (V4.1 or greater).

The OpenVMS server can make any of its file systems available to the network. A file system is a hierarchy of devices, directories, and/or files stored as a FILES-11 ODS-2 or ODS-5 on-line disk structure. File systems can include bound volumes and shadow sets.

The OpenVMS server *exports* the file systems, making them available to the network. Authorized clients *mount* the exported file systems, making them available to their users as if they were local directories and files.

Each file system is identified by the name of its mount point; that is, the name of the device or directory at the top of its hierarchy. When a client mounts a file system, it connects the mount point to a mount directory in its own file system. Through this connection, all files below the mount point on the server are available to client users as if they were below the client mount directory.

**Note:** Exported file system names cannot be longer than 14 characters. The NFS server allows NFS clients access to only the highest version of OpenVMS files.

Each client automatically converts mounted directory and file structures, contents, and names to the format required by its own operating system. For example, an OpenVMS file named:

```
USERS:[JOE_NOBODY]LOGIN.COM
```

might appear to a UNIX end user as:

```
/vmsmachine/users/joe_nobody/login.com
```

and to a Windows end user as:

```
E:\users\joe_nobody\login.com
```

**Note:** The NFS server can convert all valid UNIX, Windows, or Linux file names to valid OpenVMS names. Similarly, the server can convert those OpenVMS file names back to valid UNIX, Windows, or Linux names.

## Security

The NFS server provides two levels of security:

Access to Individual...	Description
File systems	Can be restricted to specific clients listed in mount restriction lists for those file systems, as described in the <i>Restricting Access to a Mount Point</i> section.
Directories and files	These are controlled on a per-user basis. The NFS server consults a database that maps users on NFS client systems to OpenVMS userids. When the NFS server receives an NFS file access request, it maps the client user identifier in the request to an OpenVMS userid/UIC and compares the UIC to the owner, protection mask, and any directory or file ACLs. The NFS server either grants or denies access, as described in the <i>Mapping Between Users' OpenVMS and Client Identifiers</i> section.

The NFS server considers default privileges as defined by the user's UAF entry that override OpenVMS protection codes (SYSPRV, BYPASS, READALL, and GRPPRV) when granting access. However, since UNIX clients don't understand OpenVMS privileges, the client may prevent an operation which would otherwise have been allowed. If the UNIX user `root` (uid 0) is mapped to an OpenVMS user with BYPASS privilege, the user `root` can access all files.

To get GROUP protection access to a file from UNIX clients, a user must pass both the client and the server protection check. The client check is done using the UNIX GID; the server check is done using the Group portion of the OpenVMS UIC. For GROUP access to be granted, a user must be in the same UIC group on the OpenVMS system and have the same GID on the UNIX system.

**Note:** All NFS security relies on trusting the client to provide the server with truthful authentication parameters. Nothing in the NFS protocol specification prevents a client from using fraudulent parameters to bypass the security system.

VMS DELETE access does not directly translate to NFS. In NFS, a user with WRITE access to the directory can delete a file. The NFS server implements DELETE access in the same way as NFS. With this in mind, it is important for the system manager to review protection settings on exported file systems.

# Mapping Between Users' OpenVMS and Client Identifiers

Clients identify each user to the network by a pair of UNIX or UNIX-style user-ID (UID) and group-ID (GID) codes. During an access operation, the server translates back and forth between the user's OpenVMS UIC and UID/GID pair. Whenever the server starts up, it reads the `NFS.CONFIGURATION` file which includes a UID translation database that maps each user's OpenVMS user name to their client UID/GID pair. The server translates each user name to its UIC and builds a translation table to map between each UID/GID pair and UIC.

As described in the following sections, you must create and maintain the UID translation list which maps between each users' OpenVMS user name and UID/GID pair.

**Note:** For file protections to work properly, each mapping must be both unique and consistent in each direction (see the *Grouping NFS Client Systems for UID/GID Mappings* section for a description of exceptions to this rule). You cannot map a single UID to multiple OpenVMS user names, nor can you use a single user name for multiple UIDs.

For a PC-NFSD client user, you must create a UNIX-type UID/GID pair when you specify the mapping. Whenever the user provides the correct access information to the server, the server provides the client with the user's UID/GID.

To display the current UID translation list, use the `SHOW` command described in the *Invoking the NFS Configuration Utility (NFS-CONFIG) and Displaying Configuration Information* section.

## Grouping NFS Client Systems for UID/GID Mappings

In the MultiNet UID/GID to OpenVMS user name translation database, each entry is associated with a particular NFS group. An NFS group is a collection of NFS systems that share a single set of UID/GID mappings. Within an NFS group, the mapping between UID/GID pairs and OpenVMS user names on the server system must be one-to-one. You cannot map a single UID/GID to multiple user names, nor can you use a single user name for multiple UID/GIDs. However, duplicate translations may exist between NFS groups.

If no NFS group is specified when a UID/GID translation is added to the configuration, the translation is placed in the "default" NFS group. Translations in this group are used only for client systems not specified in an NFS group.

**Note:** A client system *must not* reside in more than one NFS group.

When the NFS server receives an NFS request from a client, it consults the local NFS group database to determine which group the client is associated with. If the client is not specified explicitly in a group, it is assumed to be in the default group. Once the NFS server has determined the NFS group to which the client belongs, it uses the UID/GID translation list for that group to determine the OpenVMS user name (and hence, OpenVMS UIC) to use when accessing local files.

If there is no UID/GID mapping for a user in the NFS group to which the client system belongs, the user is treated as unknown, and the UID/GID -2/-2 is used. Any translations in the default group are *not* considered if the client is specified in an NFS group.

UNIX password files may be copied from client systems for UID/GID translations when OpenVMS user names are the same as those on the client. UNIX password files may also be placed in an NFS group. With the addition of a password file, be sure the UIDs within the NFS group remain unique.

Consider the following example. At Flowers Inc., the engineering department has a group of UNIX hosts, the sales department has a collection of PCs, and the marketing department has a mix of PCs and UNIX hosts. Each group also has its own UNIX system acting as an NFS server for the group. Unfortunately, the groups did not coordinate with each other when they assigned user names and UID/GID pairs, and none of the groups are willing to change their current configurations. The accounting department, on the other hand, recently purchased a VAX 4000 computer running OpenVMS and the NFS server and wishes to make certain personnel data available via NFS to the other groups.

The accounting system manager configures the NFS server on the VAX system as follows:

1. Using the `NFS-CONFIG ADD NFS-GROUP` command, the system manager creates the three NFS groups `ENGINEERING`, `SALES`, and `MARKETING`, placing the NFS systems in each department in the appropriate NFS group. The default group is used for NFS systems in the accounting department.
2. The system manager obtains UID/GID mappings from each department and creates OpenVMS user names for each NFS client user who needs access to the NFS server on the OpenVMS system.
3. Finally, the system manager uses the `NFS-CONFIG ADD UID-TRANSLATION` and `ADD NFS-PASSWD-FILE` commands to create the mappings between OpenVMS user names and the UID/GID pairs for each NFS group. See *Naming Mount Points* for details on specifying these mappings.

If all systems in your environment share the same UID/GID pairs, you do not need to create or specify NFS groups. All translations are automatically placed in the default group (which has no group name associated with it).

## Handling Incomplete Mappings

When mappings are incomplete or nonexistent, access operations are severely limited or denied.

If any OpenVMS files or directories are owned by a UIC for which there is no mapping, the server handles them as though they were owned by the UNIX user "nobody," whose identifiers are UID -2, GID -2. Similarly, if any client users have UIDs for which there are no mappings, the server grants them access to OpenVMS files as if they were the OpenVMS user DEFAULT, whose UIC is [200,200]. In either case, only the WORLD-READ and WORLD-EXECUTE file protection settings are granted.

WRITE access is never granted to unmapped users even in those cases where OpenVMS protections allow WORLD WRITE access.

## UNIX File System Semantics

This section describes the techniques the NFS server uses to map UNIX file system semantics to OpenVMS file system semantics.

## Mapping UNIX File Links

The NFS server provides primitive support for symbolic and hard link operations under OpenVMS.

Because the OpenVMS file system has no support for symbolic links, symbolic links created by an NFS client are stored under OpenVMS in a file with undefined record attributes which begins with \*SYMLINK\*. Using this method of storing link contents, the NFS server *appears* to support symbolic links, but the links cannot be used directly by OpenVMS applications.

The NFS server supports the hard link operation by making additional directory entries for a file under OpenVMS. These hard links can be used directly by OpenVMS applications. Unlike the UNIX file system, the OpenVMS file system does not keep a link reference count on files which have multiple links, although the NFS server attempts to simulate this by keeping a reference count in memory. After the reference count is purged by a reboot or by restarting the NFS server, deleting a file with multiple hard links could result in the loss of data although there may be other directory entries.

Likewise, deleting a file's remaining directory entry (if the file previously had hard links) could result in the contents of the file not being deleted, resulting in a lost file which can later be found by disk analysis. These limitations are inherent in the OpenVMS file system.

# Mapping UNIX Device Special Files

Device block and character special files created by an NFS client are stored under OpenVMS in a file with undefined record attributes which begins with \*SPECIAL\*. Using this method of storing special files, the NFS server *appears* to support them, but the files cannot be used directly by OpenVMS applications.

# Mapping UNIX setuid, setgid, and "sticky" File Modes

The NFS server appears to support the `setuid`, `setgid`, and sticky (VTX) file modes by using the reserved-to-customer bits in the user characteristics field of the file header. Although these protection modes have no meaning to OpenVMS, the NFS server still stores them.

# Mapping UNIX File Names

The NFS server attempts to store files with any file name, even when client file names contain characters not permitted by OpenVMS. To accomplish this, the NFS server performs a mapping between OpenVMS and NFS client file names, using the inverse mapping of the NFS client. This mapping ensures consistency between other NFS clients accessing and creating files using the NFS server, and the NFS client accessing and creating files using other NFS servers. All mapping sequences on the OpenVMS server begin with the "\$" escape character. This file name mapping can be disabled as described in the *Mount Point Option Summary* section.

As "\$" is the mapping sequence escape character, a real "\$" in a file name as seen by the client is mapped to "\$\$" on the OpenVMS server. For example, the client file name `foo$bar.c` maps to `FOO$$BAR.C` on the OpenVMS server.

A "\$" followed by a letter (A to Z) in a file name on the server indicates a case-shift in the file name on the client. For client systems like UNIX which support case-sensitive file names, a file name can begin in lowercase and change back and forth between uppercase and lowercase. For example, the client file name `"aCaseSENSITIVEfilename"` would map to `"A$C$ASE$SENSITIVEF$ILENAME"` on the OpenVMS NFS server.

A "\$" followed by any digit 4 to 9 indicates a mapping as shown in the table below.

VMS Char.	Server Char.	Hex Value	VMS Char.	Server Char.	Hex Value	VMS Char.	Server Char.	Hex Value
\$4A	^A	1	\$5A	!	21	\$7A	Space	20



\$4B	^B	2	\$5B	"	22	\$7B	;	3B
\$4C	^C	3	\$5C	#	23	\$7C	<	3C
\$4D	^D	4	\$5E	%	25	\$7D	=	3D
\$4E	^E	5	\$5F	&	26	\$7E	>	3E
\$4F	^F	6	\$5G	`	27	\$7F	?	3F
\$4G	^G	7	\$5H	(	28			
\$4H	^H	8	\$5I	)	29	\$8A	@	40
\$4I	^I	9	\$5J	*	2A	\$8B	[	5B
\$4J	^J	A	\$5K	+	2B	\$8C	\	5C
\$4K	^K	B	\$5L	'	2C	\$8D	]	5D
\$4L	^L	C	\$5N	.	2E	\$8E	^	5E
\$4M	^M	D	\$5O	/	2F			
\$4N	^N	E	\$5Z	:	3A	\$9A	`	60
\$4O	^O	F				\$9B	{	7B
\$4P	^P	10	\$6A	^@	00	\$9C		7C
\$4Q	^Q	11	\$6B	^[	1B	\$9D	}	7D
\$4R	^R	12	\$6C	^\	1C	\$9E	~	7E
\$4S	^S	13	\$6D	^]	1D	\$9F	DEL	7F
\$4T	^T	14	\$6E	^^	1E			
\$4U	^U	16	\$6F	^_	1F			
\$4V	^V	16						
\$4W	^W	17						
\$4X	^X	18						
\$4Y	^Y	19						
\$4Z	^Z	1A						

The digit after the dollar sign and the trailing letter indicates the character in the client file name. In the special case of the "dot" character (.), the first dot in the client file name maps directly to a dot in the server OpenVMS file name. Any dot characters after the first one in the client file name are mapped to the character sequence \$5N on the OpenVMS server. In directory files, any dot in the client file name maps to \$5N on the server. For example, the client file name `foo.bar#1.old` maps to `FOO.BAR$5C1$5NOLD` on the OpenVMS server unless `foo.bar#1.old` was a directory file, in which case it would map to `FOO$5NBAR$5C1$5NOLD.DIR` on the OpenVMS server.

Finally, a "\$" followed by a three-digit octal number indicates a character in the file name on the server that has the binary value of that three-digit octal number. As all character binary values from 0 to 177 (octal) already have mappings, only characters from 200 to 377 are mapped in this fashion. Thus, the leading digit of the octal number must be either 2 or 3.

## Mapping OpenVMS Text Files to UNIX Text Files

The NFS server attempts to make access to text files as transparent as possible. Most OpenVMS files containing ASCII text have RMS record attributes of Variable Length Records with Carriage Return Carriage Control (VAR-CR). When VAR-CR files are read via the NFS server, the NFS server automatically converts the contents of the file into the equivalent UNIX byte stream. Because of this conversion, there are a number of restrictions imposed on VAR-CR files:

- VAR-CR files cannot be written to unless the file is first truncated to a size of zero.
- Access to extremely large VAR-CR files (those larger than the size of the cache specified by `MAXIMUM-CACHE-BUFFERS` and `MAXIMUM-FILESYSTEM-BUFFERS`) is extremely slow, and may result in NFS timeouts if the file system is not mounted with a large enough timeout value. This happens because the NFS server may need to read the entire file to convert a small portion near the end of the file, and because the entire file must be read to determine its size when represented as a Stream file.

All other types of files are not converted by the NFS server. The client sees the raw disk blocks directly.

## NFS Server Architecture

The NFS server includes five top-level protocols that run parallel to each other over a stack of lower-level protocols. The top-level protocols are:

- The Network File System protocol (NFS) is an IP-family protocol that provides remote file system access, handling client queries.
- The RPC (Remote Procedure Call) mount protocol (RPCMOUNT) is used by clients to mount file systems and get mount-point information.

- The RPC-protocol port mapper (RPCPORTMAP) performs the mapping between RPC program numbers and UDP and TCP port numbers.
- The RPC quota daemon (RPCQUOTAD) returns disk quota information.
- The RPC status monitor (RPCSTATUS) and the RPC lock manager (RPCLOCKMGR) together coordinate access to sections of files.
- The PC-NFSD protocol provides authentication and remote-printing functions specific to PC-NFS. Only PC and PC-compatible clients use this protocol.

Underlying the NFS, RPCLOCKMGR, RPCMOUNT, RPCPORTMAP, RPCQUOTAD, RPCSTATUS, and PC-NFSD protocols is a stack of protocols:

- The Remote Procedure Call (RPC) protocol allows clients to make procedure calls across the network to the server.
- The external Data Representation (XDR) protocol handles architectural differences between the client and server systems, allowing the NFS protocol to communicate between systems of dissimilar architectures.
- The User Datagram Protocol (UDP), Transmission Control Protocol (TCP), and Internet Protocol (IP) are used for the lowest levels of communication.

Traditionally, NFS has always been run over UDP. The NFS server also supports communication over TCP, which may provide performance improvements when communicating with NFS clients across slow network links or wide area networks (WANs), which suffer from packet loss and delay.

The list of queues that PCNFSD returns in a single UDP datagram is limited to the first 45,000 bytes.

# NFS Server Configuration Overview

There are three main aspects to configuring the NFS server system:

1. Enabling the NFS server on the server host.
2. Configuring the NFS server.
3. Configuring the clients.

These operations are performed normally in the following sequence:

1. Enable the NFS server, using SERVER-CONFIG.
2. Make sure that each user who will access the server has an OpenVMS user account on the server and an account on the client.
3. Invoke NFS-CONFIG to perform Steps 4 through 6.
4. Provide the NFS server with a basis for translating between the OpenVMS and client identifiers for each user.

5. Export each file system:
  - a. Choose a name for the mount point.
  - b. Export the mount point and reload the server to make the change effective.
  - c. Mount and test the file system on each client.
  - d. If you want to restrict access to the file system to specific clients, create a mount restriction list for the mount point, restart the server, and retest the mount operation from each client.
6. *Only when necessary*, change global parameter settings (followed by a server restart), and retest the configuration. The default parameter settings are sufficient in most cases.

The following sections describe these operations.

## Enabling the NFS Server

Enable the NFS server by enabling the following services:

- NFS server
- RPCMOUNT mount server
- RPCQUOTAD quota server
- RPCLOCKMGR lock manager
- RPCSTATUS status monitor
- RPCPORTMAP RPC-protocol port mapper

For networks that include PC or PC-compatible clients with PC-NFS software, you should also enable the PC-NFSD server. Use the MultiNet Server Configuration Utility (SERVER-CONFIG) to enable these services.

The following sample sessions show how to enable protocols with SERVER-CONFIG. The first example pertains to systems that *do not* include PC clients and *do not* use the PC-NFSD protocol:

```
$ MULTINET CONFIGURE/SERVER
MultiNet Server Configuration Utility 5.6
[Reading in configuration from MULTINET:SERVICES.MASTER_SERVER]
SERVER-CONFIG>ENABLE NFS
SERVER-CONFIG>ENABLE RPCMOUNT
SERVER-CONFIG>ENABLE RPCQUOTAD
SERVER-CONFIG>ENABLE RPCPORTMAP
SERVER-CONFIG>ENABLE RPCLOCKMGR
SERVER-CONFIG>ENABLE RPCSTATUS
SERVER-CONFIG>RESTART
Configuration modified, do you want to save it first ? [YES] YES
[Writing configuration to
SYS$COMMON:[MULTINET]SERVICES.MASTER_SERVER]
```

```
%RUN-S-PROC_ID, identification of created process is 0000017A
SERVER-CONFIG>EXIT
[Configuration not modified, so no update needed]
$
```

The following example pertains to systems that do include PC clients and use PC-NFSD.

```
$ MULTINET CONFIGURE/SERVER
MultiNet Server Configuration Utility 5.6
[Reading in configuration from MULTINET:SERVICES.MASTER_SERVER]
SERVER-CONFIG>ENABLE NFS
SERVER-CONFIG>ENABLE RPCMOUNT
SERVER-CONFIG>ENABLE RPCQUOTAD
SERVER-CONFIG>ENABLE RPCPORTMAP
SERVER-CONFIG>ENABLE RPCLOCKMGR
SERVER-CONFIG>ENABLE RPCSTATUS
SERVER-CONFIG>ENABLE PCNFSD
SERVER-CONFIG>RESTART
Configuration modified, do you want to save it first ? [YES] YES
[Writing configuration to
SYS$COMMON:[MULTINET]SERVICES.MASTER_SERVER]
%RUN-S-PROC_ID, identification of created process is 0000017A
SERVER-CONFIG>EXIT
[Configuration not modified, so no update needed]
$
```

## Creating OpenVMS User Accounts for Client Users

An OpenVMS user account must exist for each client user who will have access to the OpenVMS file systems. In addition, the account must have access to those file systems.

As described in the *Creating UID/GID Mappings* section, you must also provide the server with a basis for mapping between each user's client and OpenVMS accounts.

## Invoking the NFS Configuration Utility (NFS-CONFIG) and Displaying Configuration Information

To invoke the NFS Configuration Utility (NFS-CONFIG), enter:

```
$ MULTINET CONFIGURE/NFS
```

In response, NFS-CONFIG reads its current configuration file, NFS . CONFIGURATION, as shown in the following example. All configuration operations that use NFS-CONFIG change this file.

```
$ MULTINET CONFIGURE/NFS  
MultiNet NFS Configuration Utility 5.6  
[Reading in configuration from MULTINET:NFS.CONFIGURATION]  
NFS-CONFIG>
```

You can display various information about the current configuration in the NFS . CONFIGURATION file. The following examples include these lists:

- The file system export list, which is a list of the file systems available to the network. A mount restrictions list appears next to the entry for each file system, showing the clients that can access the file system (unless all clients can access it).
- The UID/GID to OpenVMS user name translation list.
- The global parameter list, which contains the server's global parameters.

These lists and corresponding aspects of server configurations are explained in subsequent sections of this chapter.

```
NFS-CONFIG>SHOW  
Filesystem      Restrictions  
-----  
SYS$$SYSDEVICE:  brown.example.com localhost  
UID Translations: VMS Username      Unix UID      Unix GID  
                  -----  
                  JOHN             10            15  
                  BANZAI             2             40  
NFS-CONFIG>
```

The following example shows a full display:

```
NFS-CONFIG>SHOW /FULL  
Exported Filesystem "SYS$$SYSDEVICE:"  
Mounts restricted to:  
  brown.example.com  
  localhost  
UID Translations: VMS Username      Unix UID      Unix GID  
                  -----  
                  JOHN             10            15  
                  BANZAI             2             40  
Kernel-Mode NFS server.  
Kernel-Mode exceptions will cause NFS to hibernate for debugging.  
Number of RPC Transports:          100 simultaneous requests  
Size of duplicate request cache:    250 entries  
File cache timer interval:         30 seconds  
Read-Only flush age:               50000 seconds  
Read/Write flush age:              50000 seconds  
File info flush age:               1200 seconds  
Directory info flush age:          300 seconds  
File info idle flush age:           600 seconds
```

```
Directory info idle flush age:          150 seconds
Use Directory Blocking ASTs for cache consistency
Use File Blocking ASTs for cache consistency
Maximum cache files:                   3000 files
Maximum cache buffers:                  500 buffers
Maximum open channels:                  50 channels
Maximum file system files:              3000 files
Maximum file system buffers:            500 buffers
Maximum file system channels:           50 channels
Maximum Queued Removes:                 25 files
Seconds Before Writeback:               3 seconds
  Maximum Dirty Buffers:                 0 buffers (no limit)
  Maximum Write Jobs:                    0 operations (no limit)
NFS-CONFIG>
```

# Creating UID/GID Mappings

The following sections describe how to create and manipulate UID/GID mappings.

## Adding and Deleting Mappings

There are two methods for adding and deleting mappings of user names to UID/GID pairs. You can combine these methods as needed:

- Add and delete individual mappings and NFS groups with `NFS-CONFIG`.
- If the system includes UNIX clients with users with the same UNIX and OpenVMS user names, use one or more `/etc/passwd` files as the basis for multiple mappings and add those mappings to the configuration with `NFS-CONFIG`.

After creating or modifying the UID translation list, reload the server to make the changes take effect, as described in the *Reloading the NFS Server Configuration and Restarting the Server* section.

## Adding and Deleting Individual Mappings

The `ADD UID-TRANSLATION` command creates an individual mapping between an OpenVMS user name and a UID/GID pair. For example:

```
NFS-CONFIG>ADD UID-TRANSLATION JOHN 10 15
```

To create a mapping between an OpenVMS user name and a UID/GID pair associated with the NFS group `MARKETING`, for example:

```
NFS-CONFIG>ADD UID-TRANSLATION JOHN 10 15 MARKETING
```

If you are creating UID/GID pairs, each code must be a positive integer or zero, and each user must have a unique UID, independent of the operating system the client is running. Someone who uses multiple clients must have the same UID for each of the clients, or use NFS groups to group together systems sharing the same UID mappings. To delete an individual mapping, use the `DELETE UID-TRANSLATION` command:

```
NFS-CONFIG>DELETE UID-TRANSLATION JOHN
```

To delete a mapping associated with an NFS group:

```
NFS-CONFIG>DELETE UID-TRANSLATION MARKETING/JOHN
```

## Adding and Deleting NFS Groups

Use the `ADD NFS-GROUP` command to create an NFS group. For example:

```
NFS-CONFIG>ADD NFS-GROUP SALES WHORFIN.EXAMPLE.COM, CC.EXAMPLE.COM
```

**Note:** Client names must be fully qualified.

To delete a system from an NFS group, use the `DELETE NFS-GROUP` command:

```
NFS-CONFIG>DELETE NFS-GROUP SALES WHORFIN.EXAMPLE.COM
```

To delete the NFS group itself, use an asterisk (\*) for the host specification:

```
NFS-CONFIG>DELETE NFS-GROUP SALES *
```

## Adding Multiple Mappings

The `/etc/passwd` files from UNIX NFS clients can be used to create multiple mappings *only when* the user names on the UNIX and OpenVMS systems are the same. To create a multi-user mapping, use FTP (or another file transfer utility) to copy each applicable `/etc/passwd` file from the UNIX system to the OpenVMS system running the NFS server. Use the `NFS-CONFIG ADD NFS-PASSWD-FILE` command to create the mapping. For example:

```
NFS-CONFIG>ADD NFS-PASSWD-FILE MULTINET:NFS.PASSWD
```

To create a multi-user mapping associated with the NFS group `MARKETING`, you might use the command:

```
NFS-CONFIG>ADD NFS-PASSWD-FILE MULTINET:NFS.PASSWD1 MARKETING
```



**CAUTION!** If you add or delete users, or change the mapping between user name and UID/GID in an `/etc/passwd` file on an NFS client, be sure to make the same change in the NFS `passwd` file on the server.

The following example shows a UID translation list that includes both individual mappings and `passwd` file entries created with the `NFS-CONFIG ADD UID-TRANSLATION` and `ADD NFS-PASSWD-FILE` commands (excerpted from the output of a `SHOW` command).

```
NFS Passwd Files: MULTINET:NFS.PASSWD, MULTINET:NFS.PASSWD2
UID Translations: VMS Username      Unix UID      Unix GID
                   -----
                   JOHN             10            15
                   BANZAI           2              40
```

The next example shows a UID translation list that includes NFS group entries, individual mappings, and `passwd` file entries created with the `NFS-CONFIG ADD NFS-GROUP`, `ADD UID-TRANSLATION`, and `ADD NFS-PASSWD-FILE` commands (excerpted from the output of a `SHOW` command).

```
NFS Group Name      Members
-----
ENGINEERING         control.example.com, fang.example.com
SALES                small-berries.example.com, whorfin.example.com
NFS Passwd Files:  ENGINEERING/MULTINET:NFS.PASSWD, MULTINET:NFS.PASSWD2
UID Translations:  VMS Username      Unix UID      Unix GID
                   -----
                   JOHN             10            15
                   BANZAI           2              40
                   ENGINEERING/MAX  30            10
                   SALES/TOMMY      30            10
```

To delete an NFS `passwd` file entry, use the `DELETE NFS-PASSWD-FILE` command. For example:

```
NFS-CONFIG>DELETE NFS-PASSWD-FILE MULTINET:NFS.PASSWD
```

## Exporting File Systems

To make a file system available to the network, you must export it by adding the name of the file system's mount point to the NFS `.CONFIGURATION` file system export list, then reloading the NFS server. To display the current list, use the `SHOW` command. A sample list is included in the *Invoking the NFS Configuration Utility and Displaying Configuration Information* section.

**Note:** All directories accessible via NFS must have at least READ and EXECUTE access set for the desired level of access (SYSTEM, OWNER, GROUP, or WORLD). In particular, the root directory file 000000.DIR (and possibly other directories below it) must have WORLD READ and EXECUTE access set. Otherwise, users on UNIX and PC systems may not be able to access files in their directories below 000000.DIR, *even if* they own those files and directories. If the directory protections are set incorrectly, directories that have files in them may appear to be empty.

## Naming Mount Points

You must specify the names of the server mount points that are to be available to the network. The server accepts the following formats for mount point names:

- A device name (for example, DUA0 :)
- A device and directory name (for example, DUA0 : [USERS] or SYS\$SYSDEVICE : [USERS])
- A logical name (for example, SYS\$SYSDEVICE : or SYS\$SYSTEM:)

When a mount point can be specified with more than one name (for example, SYS\$SYSDEVICE :, DISK\$VAXVMSRL4 :, and DUA0 :) you can use any of them. However, note that the name you choose is also the name the client uses to access the file system when mounting it.

Although exported file systems can overlap, this practice is not recommended because information is duplicated in the NFS server cache. Overlapping information in the cache will often cause undesirable behavior by the NFS server.

For example, if you use the device name DUA0 : as a mount point, you make the entire file system available for access. If you use the directory name DUA0 : [USERS] as a mount point, you make DUA0 : [USERS . . . ] \* . \* available. If you specify both DUA0 : and DUA0 : [USERS] as mount points, the DUA0 : [USERS] mount point is redundant and will result in unexpected behavior by the NFS server.

**Note:** To export a directory that requires you to use more than 13 characters, you must create a logical name that points to the directory and export the logical name instead of the directory name.

# Adding File Systems to the Export List

Use the `NFS-CONFIG ADD EXPORT` command to export a mount point. Remember that you must reload the NFS server to make the new mount point available to the network.

**Note:** Do not export search paths.

The following example adds the file system `SYS$SYSDEVICE :` to the export list.

```
$ MULTINET CONFIGURE/NFS  
MultiNet NFS Configuration Utility 5.6  
[Reading in configuration from MULTINET:NFS.CONFIGURATION]  
NFS-CONFIG>ADD EXPORT SYS$SYSDEVICE :  
[Added new Exported file system "SYS$SYSDEVICE :"]  
[Current Exported File System set to "SYS$SYSDEVICE :"]  
NFS-CONFIG>RESTART
```

Once exported, a file system is "open" or available to all clients. You can restrict access to a mount point (as described in the *Restricting Access to a Mount Point* section); however, you should first configure the clients that will access it and test the resulting configuration before defining restrictions. Performing configuration operations in this sequence facilitates the verification of file system exports and mounts, since the server will not reject mount requests to an open file system.

Similarly, although you can change the settings of several global parameters (as described in the *Modifying NFS Server Global Parameters* section), wait until you have tested your initial configuration before making such changes.

If your network includes PC clients, you may want to configure the remote printing service of the PC-NFSD protocol (as described in the *Configuring PC-NFSD Remote Printing Service* section).

# Removing File Systems from the Export List

To remove a file system from the network, use the `NFS-CONFIG DELETE EXPORT` command. For example:

```
NFS-CONFIG>DELETE EXPORT SYS$DEVICE :
```

## Establishing Cluster-wide Aliases

MultiNet allows the system manager to declare a cluster-wide IP address serviced by a single node at any given time. If that node should fail, servicing of the cluster-wide IP address will "fail-over" to another node, allowing NFS clients to continue to access cluster disks even if the host running the NFS server crashes. For more information about creating cluster-wide IP addresses, refer to Chapter 11.

# Reloading the NFS Server Configuration and Restarting the Server

Whenever you change the server configuration, you alter the `NFS.CONFIGURATION` file. Most of the remaining procedures described in this chapter change the configuration. Before you can use a new or revised configuration, you must reload the NFS server, either from within `NFS-CONFIG` or from `DCL`.

Reloading the server involves reloading the NFS and RPCMOUNT services:

- Enter the following command to reload both protocols:

```
NFS-CONFIG>RELOAD
```

- Enter the following command from `DCL` to restart only the NFS server:

```
$ MULTINET NETCONTROL NFS RELOAD
```

- Enter the following command from `DCL` to restart only the RPCMOUNT protocol:

```
$ MULTINET NETCONTROL RPCMOUNT RELOAD
```

You may also restart the NFS server by killing the current process and running a new one. The following `DCL` command does this:

```
NFS-CONFIG>RESTART
```

Restarting the server causes the file cache to be flushed and the new server will need to rebuild it. Therefore, it is recommended that you use the `RELOAD` command whenever possible.

## Shutting Down the NFS Server

You can edit your `SYSHUTDOWN.COM` procedure to include commands that stop the NFS server. For example:

```
$ MULTINET NETCONTROL NFS SHUTDOWN
```

# Testing the System Configuration

Test the configuration at these times:

- After your initial configuration, when you have:
  - Specified the mappings between UIDs/GIDs and user names
  - Configured the NFS server
  - Restarted the NFS server
  - Configured one or more clients for the NFS server
- After you modify the configuration by reconfiguring the NFS server, adding clients, or reconfiguring existing clients

To test a configuration, check all file systems from one client, and at least one file system from every client:

1. Log in as one of the client's users. For example, on a Linux host client, you might log in as "joebob" (be sure your system includes a mapping for "joebob's" UID/GID and a user name on the server system).
2. Mount a file system the user can access. For instructions on mounting file systems, see the *Configuring Clients* section.
3. Check the mount as described in the next steps.
  - a. Check the contents of the file system's mount directory. For example, on a Sun host client, use the `cd` command to change to the mount directory, and the `ls -l` command to list the names of the directory's files.
  - b. Verify that files in the mount directory can be read. For example, on a Sun host client, use the `cp` command to copy a file from directories under the mount point to the `/dev/null` directory.
  - c. Verify that files can be written to the OpenVMS server. For example, on a Sun host client, use the following command to copy a file to the current directory:

```
$ cp /vmunix .
```

**Note:** Process Software recommends using the `cp` utility to test the server because it is better at reporting protection problems than most other UNIX utilities, including `cat`.

4. Repeat this process until you have mounted and checked all file systems that the client's users wish to access.

5. Log in from each of the other clients and check file system mounts as described in Steps 1 through 4.

## Checking for Errors

After exporting file systems and restarting the server, *but before configuring clients*, enter the following command:

```
$ REPLY/ENABLE=NETWORK/TEMP
```

This command causes network event messages to be displayed on your terminal, including error messages from the NFS and RPCMOUNT servers. See the *MultiNet Messages, Logicals, and DECnet Applications* book for lists of error messages and the conditions that generate them.

## Configuring Clients

After configuring the NFS server, you must configure each client that will access OpenVMS server file systems. Different types of clients require different configuration procedures. There are, however, two general guidelines for configuring clients:

- Each client must explicitly mount each file system to which it requires access. For most types of clients, a mount directory must be created for each file system.
- For most types of clients, you must change the default settings of some configuration parameters in the client's MOUNT command. These settings control how the client accesses the OpenVMS server.

The following section explains how to configure hosts running UNIX. Chapter 28 explains how to configure OpenVMS systems as clients using the NFS client software.

## Configuring UNIX Host Clients

As part of the configuration process, you must log into each client and mount all file systems the client will have access to. For each client, you may also need to adjust the `wsize`, `timeo`, and `retrans` parameters for the client's mount command. Before a file system is mounted, you must also ensure that the directory under which a file system will be mounted exists on the client.

## Mounting File Systems on UNIX Hosts

Mount each file system as follows:

1. Use the `mkdir` command to create the mount directories. For example, enter the following command while logged in as `root` to create a directory called `/mnt`:

```
# mkdir /mnt
```

2. Mount each file system by executing the mount command with conservative values for `wsiz`, `timeo`, and `retrans`, using this syntax:

```
# mount -o options file-system mount-point, retrans=5 \  
vmsmachine:sys\<$sysdevice: /mnt
```

For example:

```
# mount -o soft,rw,timeo=50, retrans=5 vmsmachine:sys\<$sysdevice: /mnt
```

Once you have mounted the remote file system, you can experiment with other `wsiz`, `timeo`, and `retrans` values to improve performance, as described in the *Explicitly Specifying Mount Parameter Settings* section.

**Note:** When the mount point name is specified with OpenVMS syntax, any special characters (for example, `$`, `[`, and `]`) must be delimited with a backslash (`\`) for proper processing by the UNIX shell.

3. If a mount is not successful, errors may be reported to the user's display or to the OpenVMS console via OPCOM. (NUL characters no longer appear in the OPCOM output.)
4. After performing a successful mount, and after adjusting the `wsiz`, `timeo`, and `retrans` values, add the file system and its mount parameters to the client's `/etc/fstab` file so file system mounts will occur automatically.

## Explicitly Specifying Mount Parameter Settings

As part of configuring a UNIX host client, you may need to change the number of block I/O daemon (`bi`od) processes or the values of one or more mount-parameter settings to correct the two problems discussed next. Make these corrections *after* performing the first mount of a file system as described in Step 2 in the procedure in the preceding section.

The OpenVMS XQP is relatively slow. There are times when the NFS server must perform many operations before returning the answer to a seemingly simple query. The resulting delay can cause a client to report "RPC timeout errors" and unnecessarily retransmit its query.

For example, accessing a large directory file can cause an unexpected delay in processing an NFS request. Process Software recommends you keep fewer than 1,000 files in each directory, especially when you frequently add and delete files.

Such problems usually occur sporadically, and are often not reproducible because the server has cached the result and can answer the query quickly when it is made a second time.

In Step 2 of *Mounting File Systems on UNIX Hosts* the problem was avoided by mounting the file system with larger than normal `timeo` (timeout) and/or `retrans` (retransmission) parameter settings. The higher `timeo` value increases the length of delay the server will tolerate before timing out. However, if a packet is lost during transmission, a large `timeo` value means a long delay before retransmission.

The higher `retrans` value increases the number and rate of retransmissions a client makes before timing out, hence decreasing the delay between retransmissions. Retransmissions do not adversely affect the server, however, as each new request is recorded in the duplicate-requests cache (described in the *Modifying NFS Server Global Parameters* section). The server discards all retransmissions (which are duplicates of the original request) as it processes the original.

The `timeo` and `retrans` values can be adjusted to achieve an appropriate tradeoff for your network. A high `timeo` value with a low `retrans` value is an appropriate solution for a reliable network that requires few retransmissions. In contrast, although specifying a high `retrans` value and a low `timeo` value can create significant overhead in unnecessary queries, this solution is appropriate for an unreliable network because it minimizes the delay when a packet is lost.

The total time available to the server to complete an operation is the product of the `timeo` and `retrans` values. For most systems, appropriate values are 50 for `timeo` (5 seconds-`timeo` is usually specified in tenths of seconds), and 5 for `retrans`.

## Restricting Access to a Mount Point

By default, when you export a file system, its mount point is "open" and available for mounting by any client on your network. However, for each exported file system, you can create a list of clients permitted to mount it. This list, called the *mount restriction list*, appears next to the name of the file system's mount point in the export list. The presence of a mount restriction list prevents all unlisted clients from mounting the mount point.

You can export a mount point for *read-only access* using the `NFS-CONFIG` command `ADD MOUNT-RESTRICTION`. Use the `-ro` (read-only) keyword instead of the `nfs_group` name. Any attempts to write to the disk specified by this mount point fails. This restriction affects any NFS group associated with that particular mount point. This example shows how to export a disk to restrict all users to read-only access:

```
NFS-CONFIG>ADD MOUNT-RESTRICTION DISK$ONE -ro
```



The next example shows how to restrict one group of users (those on BOOTE.EXAMPLE.COM) to read-only access, at the same time denying access to everyone else:

```
NFS-CONFIG>ADD MOUNT-RESTRICTION DISK$USERS BOOTE.EXAMPLE.COM
NFS-CONFIG>ADD MOUNT-RESTRICTION DISK$USERS -ro
```

Use NFS-CONFIG to create and modify mount restriction lists. Use the following procedure to create a mount restriction list for a file system's mount point or add a client to the list. You must reload the server before a new or changed list goes into effect.

1. Select the mount point by entering SELECT.
2. Add a client to the mount point's list by entering ADD MOUNT-RESTRICTION. (If no list exists, specify the first client to automatically create a list.)
3. In addition to a client name, you can specify the name of an NFS group as described in the *Grouping NFS Client Systems for UID/GID Mappings* section. Specifying a group name is equivalent to individually listing each of the clients in that group.

The following example shows the client SALES.EXAMPLE.COM being added to the mount restriction list for the SYS\$SYSDEVICE: mount point.

```
NFS-CONFIG>SELECT SYS$SYSDEVICE:
[Current Exported File System set to "SYS$SYSDEVICE:"]
NFS-CONFIG>ADD MOUNT-RESTRICTION SALES.EXAMPLE.COM
[Added Mount restriction to "SYS$SYSDEVICE:" allowing host
"sales.example.com"]
NFS-CONFIG>RESTART
$
```

4. To remove a client from the mount restriction list, use the DELETE MOUNT-RESTRICTION command followed by the RESTART command.
5. To display the mount restriction list for a mount point, use the SHOW command.

## Controlling NFS File Access with OpenVMS Access Control Lists (ACLs)

Because of the differences between OpenVMS security and UNIX system security (after which NFS is modeled), configuring the NFS server to handle ACLs properly requires a thorough understanding of both systems.

The NFS server handles ACL mappings by allowing the addition of VMS rights identifiers to the NFS UID/GID translation table. The syntax is the same as that for adding a user name to the table.

Some effort is required to determine the necessary correlation between UIC groups and rights identifiers in OpenVMS and GIDs on the NFS client. The network administrator must scan the owners and ACLs

of the files being exported and make sure all UICs and rights identifiers associated with the file system have a valid UID/GID translation. This is necessary to make sure the NFS server's representation of security information on all files is accurate.

Although file access is determined by the server based on the user's UIC, the correct representation of security information to the client can be critical. Many multi-user clients grant access to data in the cache locally, without making a request to the server. Therefore, it is imperative that the representation of a file's protection and security information is accurate.

Single-user clients do not usually have this problem. However, on any client that denies access based on returned security information, improper mapping may deny access unnecessarily.

To make sure the NFS server can handle requests for files with ACLs:

1. Make sure there are UID/GID translations for all rights identifiers in all ACLs associated with exported files.
2. For each rights identifier, make sure the appropriate users on NFS client systems have the same GID.

To illustrate this solution, consider an environment in which the files belonging to a project are exported as part of a single file system and you need to control access to each project's files by an ACL. Perform the following tasks for each project:

1. On the NFS client, select a GID for the project members.
2. Then, on the NFS server:
  - a. Create UID/GID mappings for each NFS client user who needs to access the project files (see the *Creating UID/GID Mappings* section). These mappings must match the GIDs on the client for the project.
  - b. Use `AUTHORIZE` to create a new identifier for the project.
  - c. Add a UID mapping for the project identifier. The GID associated with the project identifier must be the same as the project GID assigned to the NFS client. The choice of UID can be arbitrary, but the UID must not conflict with any other currently assigned UIDs.
  - d. Modify the protection of the project files to allow no `WORLD` or `GROUP` access. If the OpenVMS group is significant, however, you may want to allow `GROUP` access.
  - e. Add ACLs to the project files and directories that grant `READ` and `WRITE` access to holders of the project identifier that you created in Step 3.
  - f. Use `AUTHORIZE` to grant the project identifier you created in Step 2 to users with the project GID (created in Step 1).

3. Now, to add new users to the project:

- Assign the project GID to the user on the NFS client. This mapping must match the GID on the client for the project.
- Add a UID/GID mapping for the user on the NFS server.
- Grant the OpenVMS group identifier to the new user.

**Note:** The preceding procedure is the supported method for using ACLs to control access to files exported via NFS. If you cannot use this method, refer to the *How the NFS Server Interprets ACL and UIC Protection* section for details on how the NFS server converts UIC and ACL protection information into UID/GID-style file protection masks for NFS clients.

## Idiosyncrasies of ACL Support over NFS

When using ACLs, OpenVMS lets the NFS server assign different access masks for many different groups of users. When a file's attributes are transmitted to the client, the NFS protocol only lets the server return an access mask for the owner's GID; the protocol does not allow the NFS server to return multiple GIDs and their associated access masks. Because some NFS clients grant or deny access based on the protections returned with the file's attributes, the NFS server's responses to attribute requests sometimes *change the owner's GID and associated access mask* to properly represent access for the client user.

One anomaly these dynamic responses produce is that a directory listing on the client (for example, an `ls -l` command on a UNIX client) shows files accessed through ACLs as being owned by different GIDs at different times, depending on who accessed them most recently.

If the client grants or denies access based on the protection information in the cache, users may experience intermittent access failures when more than one user tries to gain access to the same file via an ACL. This phenomenon happens when the user would normally receive access through the group or through an ACE (access control list entry).

While world access can always be consistently mapped, owner access is only consistently mapped if the ACL does not contain ACEs that cannot be mapped to a GID. For details, see the *How the NFS Server Handles ACLs* section. If the UID/GID translation table is configured correctly, users should never have access to files to which they have no legitimate access on the server. However, they may intermittently be denied access.

# How the NFS Server Interprets ACL and UIC Protection

The main difficulty facing NFS server administrators is how to coordinate NFS use of the UNIX-style UID/GID protection model with OpenVMS ACL support.

**Note:** Consulting ACLs as part of an NFS server's access-checking scheme is necessary, but not sufficient, to adequately support the presence of ACLs assigned to files.

Consider the case where an OpenVMS system manager wants to grant access to files based on project groups *without* having to make sure that all client UIDs map to the same OpenVMS group. A single user may be a member of several projects, a concept incompatible with the single-group model used by VMS.

It might seem that the system manager only needs to add rights identifiers to the mapped accounts and then set up ACLs on the appropriate files. The problem with this scheme is that file protections would normally be set to `WORLD=NOACCESS`, allowing file accesses to be granted only by the ACL. However, because the file protections deny access on a UID basis, any local access checks performed at the client will fail, bypassing the ACLs.

This problem can be solved if the NFS server makes intelligent use of the NFS GID. The NFS protocol allows a single user to be identified with up to 10 groups (projects), consistent with UNIX. In this model, the NFS client checks the list of GIDs assigned to the local user to see if it matches the group ID associated with the file. If there is a match, the `GROUP` field of the file protection mask is used to determine accessibility.

The NFS server takes advantage of this model by selectively modifying the returned group ownership for files based upon applicable ACEs. The NFS server processes ACLs in the following manner. To determine whether the NFS server will grant access:

1. The NFS server obtains rights identifiers for the OpenVMS account associated with the requester's UID.
2. The NFS server selects the first (if any) ACE assigned to the file (matching one of the rights identifiers held by the OpenVMS account). The protection specified in the ACE is used in place of the protection mask associated with the file.
3. If there are no matching ACEs, the NFS server performs the standard UIC protection check.

When asked by the NFS client for the protection mask and ownership for a file, the NFS server does the following:

1. The NFS server obtains rights identifiers for the OpenVMS account associated with the requester's UID.
2. If one of the ACE identifiers matches the file owner's UIC, the NFS server uses the protection mask in the ACE to calculate the OWNER field of the protection mask returned to the NFS client. Otherwise, the NFS server uses the OWNER field of the protection mask associated with the file to calculate the OWNER field returned to the NFS client.
3. The NFS server selects the first (if any) protection ACE assigned to the file (matching one of the rights identifiers held by the requester's VMS account).
4. If the NFS server encounters a matching ACE whose identification is a UIC, and the identifier:
  - Is in the same OpenVMS group as the file owner, the server ignores the ACE.
  - Is not in the same OpenVMS group as the file owner, the server maps the requester's UIC and GID along with the ACE's protection mask as the owner of the file when returning NFS attribute information
5. If the NFS server selects an ACE, the group ownership it returns to the NFS client is taken from the GID associated with the matching identifier.

If no matching ACE is found, the NFS server obtains the GID for the file from the file owner.

To assign GIDs to UICs and identifiers, use the `NFS-CONFIG ADD UID` command described in the [Creating UID/GID Mappings](#) section.

Under this scheme, the system manager sets file protections as needed (for example, `W:NOACCESS`, `G:RWE`), and creates an ACL to grant access to processes holding a specific rights identifier.

When the NFS client performs local access checking, it compares the list of GIDs (associated with the user) against the file's group ownership, which the NFS server bases on the ACL information for the file. This scheme prevents the client's caching mechanism from defeating the ACLs associated with the file.

## How the NFS Server Handles ACLs

The key to understanding how ACLs affect file access is in the exchange that takes place when an NFS client requests attributes for a file or directory it wants to access. The client sends the server the user's UID/GID pair when it identifies the file it wants to access. The server must respond with the UID/GID pair of the file's owner along with the protections on that file in UNIX format (R/W/E for owner, group, and world/others). To accomplish this, the NFS server must translate the OpenVMS protection mask,

applicable ACEs, and UIC-based file owner into a UNIX-style protection mask and UID/GID-based owner.

If the file being requested has no ACLs associated with it, the NFS server simply returns the OpenVMS file owner's UID/GID pair which it obtains from the NFS server's UID translation table and the file's owner, group, and world protections.

If the file has an ACL, the NFS server scans the ACL for ACEs in a format that does not allow the NFS server to map group protections. These ACLs must be handled in a special way (see the *Handling ACLs with Unmappable ACEs* section).

If there are no unmappable ACEs, the client's UID is translated, and the ACL is scanned for a match based on the associated UIC. At the same time, the list is also scanned for ACEs that should be mapped to world or owner protections. Based on the scan, the server returns attributes as follows:

- The OWNER protection mask returned is the owner default protection mask logically OR'd with the access mask of the first ACE matching the owner's UIC and associated rights identifiers. This emulates OpenVMS behavior and prevents the owner of the file from being denied access because of an ACE.
- The WORLD protection mask returned is the access mask associated with the first "wildcard" ACE, if one exists. Otherwise, the WORLD protection mask returned is the default WORLD protection.
- The GROUP protection mask returned is the access mask associated with the first ACE matching the requestor's UIC and associated rights identifiers. The GID returned is the GID translation of the rights identifier or UIC that matched this ACE. If no such ACE is found, the GROUP protection mask returned is the default group protection mask and the GID returned is the GID translation of the file's owner.
- The UID returned is the UID translation of the file's owner.

## Handling ACLs with Unmappable ACEs

Occasionally, ACE access cannot be mapped to a GID as described in the previous section. This happens when the ACE identifier is specified in the following manner:

[ \*, member ]

This also happens in cases of multiple identifiers on a single ACE, such as:

ACE	Description
A+B	A and B represent rights identifiers.

<code>[a,*]+[b,*]</code>	a and b represent UIC groups.
<code>A+[a,*]</code>	A is a rights identifier and a is a UIC group.

If the ACL associated with the file contains any ACEs that cannot be mapped to a GID, file attributes are returned as follows:

- The owner protection mask returned is the access mask associated with the first ACE matching the requestor's UIC and associated rights identifiers. If no such ACE is found, the owner protection mask returned is the default protection mask appropriate for the requestor; that is, the owner's default protection mask if they own the file, the group protection mask if appropriate, and so on.
- The owner UID/GID returned is the UID/GID translation of the requestor.
- The group protection mask returned is NONE.
- The world protection mask returned is NONE.

The NFS server cannot accurately represent OpenVMS protections in this case. This technique ensures no users are granted access to data to which they would not normally have access in the client's cache on the server. However, on multi-user clients where access is denied based on cached file attributes, this mapping may result in intermittent access failures to other users trying to access the file simultaneously.

## Disabling the NFS Server's ACL Support

You can disable the NFS server's ACL support with the logical name `MULTINET_NFS_SERVER_NFS_ACL_SUPPORT_DISABLED`. To do so, this logical name must be defined as `TRUE` or `YES` in the system-wide logical name table.

After defining the logical name, restart the NFS server so the definition takes effect.

## `/VMS_STYLE_CREATE` Mount Point Option

The `/VMS_STYLE_CREATE` mount point option instructs the NFS server to use OpenVMS semantics to determine the file owner when a file is created by an NFS client. This mechanism lets NFS clients create files and charge disk quota to a rights identifier in the same way that OpenVMS users are accustomed.

Normally, the NFS server sets the file owner field exactly as specified by the NFS client software.

## Limitations and Restrictions

Alarm ACEs are not supported.

# Configuring PC-NFSD Remote Printing Service

The MultiNet PC-NFSD server includes support for a remote printing service used by PC and PC-compatible clients running PC-NFS. This section describes how to configure the service.

Before you can configure the service, the PC-NFSD server must be enabled (usually performed when enabling the NFS server). Before PC users can use the service, you must set up OpenVMS accounts, UID/GID names, and UID/GID-user name translations for them. The cache-interrupt parameters (described later in the *Modifying NFS Server Global Parameters* section,) must also be set to 1 (the default setting).

To configure the service, you must specify a generic mount directory name the server can use to create individual mount directories for spool areas on the clients. Each of these directories must be below an exported mount point available to all of the PCs.

The exported mount point directory must allow write access by the PC client users so their subdirectories and print files can be created.

For example, if the PC-NFSD spool area is specified as `SYS$SYSDEVICE:[TMP]`, all PCNFS user print files are placed in this directory, and all PCNFS users must have write access to the `SYS$SYSDEVICE:[TMP]` directory.

If the PC-NFSD spool area is specified as `SYS$SYSDEVICE:[TMP.%]`, PCNFS user print files are placed in individual subdirectories of `SYS$SYSDEVICE:[TMP]`, and all PCNFS users must have write access to the `SYS$SYSDEVICE:[TMP]` directory.

The generic name can include a percent (%) character, which is replaced by the names of individual clients when the server creates their spool directories.

For example, if all clients can access either `SYS$SYSDEVICE:` or `SYS$SYSDEVICE:[TMP]` exported file systems, the name `SYS$SYSDEVICE:[TMP.%]` could be used for the generic spool directory. When the server receives a client's first remote printing request, the server creates a mount spool directory for that client. The server defines the name for this directory by using the generic directory name and replaces "%" with the client's name. The server supplies the name to the client, and



the client mounts the directory. From then on, the client uses the directory to hold all files to be printed remotely, and the server performs all of the printing operations.

Configure the remote-printing service as described in the following example, and as illustrated in the following example.

**Note:** You use `SERVER-CONFIG` (*not* `NFS-CONFIG`) for this task. You also use `SERVER-CONFIG` to restart the server after you have finished.

1. Check the following settings before configuring the remote printing service:

- Make sure file protections on the exported directories and subdirectories are as desired for the specified user names. Make sure file protection on the exported directory allows at least `WORLD:RE` access.
- For PCNFS printing, make sure the PC-NFSD spool directory allows write access by the user and the user has sufficient disk quota if disk quotas are enabled on the OpenVMS volume that contains the PC-NFSD spool area.

2. Invoke the `SERVER-CONFIG` utility:

```
$ MULTINET CONFIGURE/SERVER
```

3. Select the PC-NFSD protocol:

```
SERVER-CONFIG> SELECT PCNFSD
```

4. Invoke the parameter-editing procedure:

```
SERVER-CONFIG> SET PARAMETER
```

5. If the `SPOOL-DIRECTORY` parameter is set, the utility asks you if you want to delete it. Respond by entering YES.

```
Delete parameter "spool-directory" ? [NO] Y
```

6. The utility prompts you to add parameters or exit the utility. Set the `SPOOL-DIRECTORY` parameter and specify the generic directory name for it:

```
You can now add new parameters for PCNFSD. An empty line terminates.
```

```
Add Parameter: SPOOL-DIRECTORY directory-name
```

*directory-name* is the generic spool directory name.

7. In response, the utility prompts you again to add parameters or exit the utility. Press **RETURN** to exit.

8. Restart the MultiNet master server.

```

$ MULTINET CONFIGURE/SERVER
MultiNet Server Configuration Utility 5.6
[Reading in configuration from MULTINET:SERVICES.MASTER_SERVER]
SERVER-CONFIG>SELECT PCNFSD
[The Selected SERVER entry is now PCNFSD]
SERVER-CONFIG>SET PARAMETER
Delete parameter "spool-directory" ? [NO] Y
You can now add new parameters for PCNFSD. An empty line terminates.
Add Parameter: SPOOL-DIRECTORY SYS$SYSDEVICE:[TMP.%]
Add Parameter:
[Service specific parameters for PCNFSD changed]
Restart the server to make these changes take effect.
SERVER-CONFIG>RESTART
Configuration modified, do you want to save it first ? [YES] Y
[Writing configuration to
SYS$COMMON:[MULTINET]SERVICES.MASTER_SERVER]
%RUN-S-PROC_ID, identification of created process is 000002CD
SERVER-CONFIG>EXIT
$

```

Use the logical name MULTINET\_PCNFSD\_QUEUE\_TYPES to select the type of queues you want returned. Define the queues to be a comma-separated list of these valid queue types: GENERIC, PRINTER, SERVER, SYMBIONT, and TERMINAL.

```

$ DEFINE/SYSTEM/EXECUTIVE MULTINET_PCNFSD_QUEUE_TYPES "PRINTER"

```

Use the logical name MULTINET\_PCNFSD\_PRINTER\_LIMIT to determine if the returned packet size is to be limited (takes a number for its value, in bytes). If this logical is not defined, MultiNet determines the size of the packet at run-time. For example:

```

$ DEFINE/SYSTEM/EXECUTIVE MULTINET_PCNFSD_PRINTER_LIMIT 45000

```

## Modifying NFS Server Mount Point Options

By default, the NFS server maps OpenVMS file system semantics to UNIX file system semantics. File names undergo a special mapping and only the top version of files is accessible through NFS. This section describes the mount point options that control or disable this conversion.

The NFS protocol specification requires the NFS server to act like a UNIX file system. If you use any of the options described in this section, the NFS server acts more like an OpenVMS file system, and may be incompatible with some NFS clients.

The mount point options are specified as "qualifiers" to the mount point name, separated from the mount point name with a "#" character. With the NFS client, the switches are part of the directory specification

being mounted. Because the switches are passed in the name, these options cannot be used with the automounter.

## Mount Point Option Summary

The below table shows the qualifiers used to control the behavior of the NFS server.

Qualifier	Description
/APPROXIMATE_TEXT_SIZE	Allows UNIX commands such as <code>ls -l</code> to execute faster by determining file sizes only when the OpenVMS file length exceeds the specified threshold.
/CREATE_NEW_VERSION	Causes an NFS <code>create()</code> operation on an already existing file to create a new version of the file instead of overwriting the old version. This qualifier has no effect on ULTRIX clients, because they do not send the correct NFS operation.
/DISPLAY_TOP_VERSION	Causes the NFS server to display the OpenVMS version number at the end of a file name when that file name is the highest version number available. The version number is usually not displayed.
/DISPLAY_VERSION	Causes the NFS server to display files that are not the highest version number. These files are usually not displayed.
/VMS_FILENAMES	Disables the file name mapping described in the <i>Mapping UNIX File Names</i> section.
/VMS_LOWERCASE_FILENAMES	Disables the file name mapping described in the <i>Mapping UNIX File Names</i> section, but changes the OpenVMS name to lowercase for display.

**Note:** This applies to ODS-2 exports only.

<code>/VMS_STYLE_CREATE</code>	Enables the use of OpenVMS semantics instead of NFS semantics for determining ownership of created files. With NFS semantics, the NFS client specifies everything explicitly. With OpenVMS semantics, file ownership may be inferred from the parent directory, ACLs, previous versions, and so on.
--------------------------------	---

## Examples of Mount Point Option Usage

The following example shows mounting and accessing a file system from UNIX using mount point options.

```
# mount -o soft,rw kaos:/users\#/vms_filenames/display_version /mnt
# ls /mnt/SMITH.DIR
BIN.DIR                TMP.DIR
LOGIN.COM              TODO
LOGIN.COM; 32          TODO.; 508
MAIL.MAI
#
```

## Modifying NFS Server Global Parameters

Global parameters affect NFS server operations. Their default settings are appropriate for almost all configurations. The following sections describe the NFS global parameters and explain how to change their values.

Global parameters can be set with the `NFS-CONFIG SET` command. For a complete list of `SET` commands, refer to the *MultiNet Administrator's Reference*. Descriptions of the global parameters are also available online with the `NFS-CONFIG HELP` command.

**Note:** Change the settings only if absolutely necessary.

The NFS global parameters control:

- Operations of the directory and file cache, including its size and discard rate

- Operations of the duplicate-requests cache
- Special operations for debugging the server

Most of the parameters control the operation of the directory and file cache that exists between the server's file systems and the network. The following sections describe the cache and the parameters.

If you must change the settings, wait until after you complete the initial system configuration and test. It is much easier to test and debug specific aspects of the NFS server before you have changed global characteristics.

## NFS Mode of Operation

The default for the server is kernel mode. If the server becomes unresponsive, reboot the server and change to user mode. This will, however, give you slower performance. If the server becomes unresponsive in user mode, issue this command

```
$ STOP /ID=pid.
```

If the server crashes when in user mode, restart the server. The commands for changing to the two modes follow:

- To put into user mode:

```
$ mu conf/nfs
NFS-CONFIG>SET USER-MODE 1
[Global NFS parameter "user-mode-server" set to 1]
NFS-CONFIG>restart
Configuration modified, do you want to save it first ? [YES] YES
[Writing NFS file server configuration to
MULTINET_COMMON_ROOT:[MULTINET]NFS.CONFIGURATION]
Connected to NETCONTROL server on "127.0.0.1"
< pc4.example.net Network Control V5.6 at Thu 26-Oct-2019 4:29PM-EST
< NFS/RPCLockMgr Server Started
< RPCMOUNT database reloaded
NFS Client UID mappings reloaded.
NFS-CONFIG>exit
```

- To put into kernel mode (default):

```
NFS-CONFIG>SET USER-MODE 0
[Global NFS parameter "user-mode-server" set to 0]
NFS-CONFIG>restart
Configuration modified, do you want to save it first ? [YES] YES
[Writing NFS file server configuration to
MULTINET_COMMON_ROOT:[MULTINET]NFS.CONFIGURATION]
Connected to NETCONTROL server on "127.0.0.1"
< pc4.example.net Network Control V5.6 at Thu 26-Oct-2019 4:29PM-EST
< NFS/RPCLockMgr Server Started
```

```
< RPCMOUNT database reloaded
NFS Client UID mappings reloaded.
NFS-CONFIG>exit
```

# NFS Server Memory Considerations

The NFS server uses memory in the following manner to cache files and directories for faster access, to hold various internal states, and to buffer requests that arrive from the clients:

- The NFS code and fixed data structures require about 1400 pages.
- The server's file and directory cache consumes the most memory-by default, a little under 20,000 pages (or 10 megabytes) of virtual memory.

As you install and configure OpenVMS and the NFS server, make sure that no OpenVMS limitation will interfere with these server requirements. If possible, specify a value of at least 30,000 for the `SYSGEN VIRTUALPAGECNT` parameter, and provide at least 30,000 pages of space in the system page file. If these resources are not available, adjust the settings of the server's parameters to decrease the maximum size of the cache, allowing it to fit within the limits of the available memory.

The following equation defines the maximum amount of memory the server can use at one time, as a function of the global parameters.

$$\text{total\_memory\_consumption} = \text{fixed\_consumption} + \text{variable\_cache\_consumption}$$

<code>fixed_consumption</code>	= 1400 pages
<code>variable_cache_consumption</code>	= (1 page x <code>MAXIMUM-CACHE-FILES</code> ) + (17 pages x <code>MAXIMUM-CACHE-BUFFERS</code> ) + (1 page x <code>MAXIMUM-OPEN-CHANNELS</code> )  <code>MAXIMUM-CACHE-FILES</code> - is the maximum number of file headers that can be cached.  <code>MAXIMUM-CACHE-BUFFERS</code> - is the maximum number of data buffers that can be cached.

	MAXIMUM-OPEN-CHANNELS - is the maximum number of channels that can be open at a time between the disk and the cache.
--	--

## Process Memory

Process memory requirements and limitations are defined by:

- SYSGEN parameters you set when configuring OpenVMS
- NFS server process quotas
- NFS server global parameters you specify when configuring the server

## Virtual and Physical Memory

OpenVMS imposes two memory limits: *virtual and physical*. As indicated in the following discussion, you must make sure that OpenVMS provides the server with adequate resources for both virtual and physical memory and that the relationship between the amounts of the two is appropriate.

The amount of virtual memory available to the server is defined by the lesser of:

- The SYSGEN VIRTUALPAGECNT parameter
- The NFS server PAGEFILE quota-by default, 65,536 pages

The amount of physical memory available to the NFS server is defined by the lesser of:

- The SYSGEN WSMAX parameter
- The NFS server WSQUOTA and WSEXTENT quotas-by default, 2,000 and 20,000 pages, respectively

When you install and configure an OpenVMS server, be sure to provide the server with enough virtual and physical memory. If the server runs out of virtual memory, it returns ENOBUFS error messages to clients whose requests cannot be satisfied. See the *MultiNet Messages, Logicals, and DECnet Applications* book for information about ENOBUFS messages.

It is equally important to provide enough physical memory to the server to prevent excessive page faulting under normal operation. If physical memory is scarce, reduce the cache's default size so enough memory is available to hold it without heavy page faulting, or allow the page faulting.

In general, a disk read performed to satisfy a page fault requires far fewer resources than a disk read performed to replace part of the cache that has been removed. This removal occurs as the server reaches

the cache size limit specified in the configuration parameters. However, during a page fault, the server can perform no other activity for any client.

Under conditions of high load from many clients, better performance usually results from reducing the size of the cache to eliminate or reduce page faulting. Under high load from a few clients, better performance usually results from allowing the page faulting.

## OpenVMS Channels

When a client requests information not in the cache, the server uses OpenVMS channels to access the required directories and files from disk. When you install and configure an OpenVMS server, you must ensure enough channels are available for server requirements. If the server runs out of OpenVMS channels, it returns an `ENOBUFS` error message to the client that requested the additional channel.

By default, the server can use up to 50 channels at once. (This number is appropriate for almost all systems, because channels are generally deassigned shortly after they have been used to read data into the cache.) You can increase or decrease the maximum number of channels by adjusting the setting of the server's `MAXIMUM-OPEN-CHANNELS` global parameter.

If you plan to increase the `MAXIMUM-OPEN-CHANNELS` value, you might need to increase the setting of the `SYSGEN CHANNELCNT` parameter when you install the server because this OpenVMS parameter must always have a value that is at least 10% greater than that of `MAXIMUM-OPEN-CHANNELS`. Do not increase the value for `MAXIMUM-OPEN-CHANNELS` above 450; this limit is set by the server's open file limit (`FILLM`) quota.

For more information about the global parameters that affect channel availability and operation, see the *Directory and File Cache Parameters* section.

## Directory and File Cache Parameters

The directory and file cache holds data from directories and files that has been requested by client users. When answering repeated requests for the same data, the server uses the cache rather than the disk, greatly improving response time.

## Channels, File Headers, and Data Buffers

When an NFS client user first requests information about a directory or file, the server assigns a channel to access it, and creates a cache entry to hold the contents of a file header. A cached file header contains information about characteristics of the directory or file (for example, its size or owner).



As the user requests data from the directory or file, the server creates 8-kilobyte data buffers for it, using the channel to read the data from disk.

If a channel remains inactive for a pre-specified length of time, the server deassigns the channel. However, the cached header and data buffers remain in the cache, and user requests can be satisfied without accessing the file on disk again.

## Directory and File Times

OpenVMS does not correctly update modification dates for directories on disk. (A directory modification date is the time when the specified directory was brought into the NFS server's directory and file cache.) However, because clients rely on modification dates when they use their own caches, the NFS server provides clients with modification dates.

## Concurrency Parameters

The `NUMBER-OF-RPC-TRANSPORTS` parameter controls the number of simultaneous requests the NFS server can process.

When the set limit is reached, no new requests are processed until one of the requests in progress completes. Processing multiple requests simultaneously prevents a single client from locking out other clients while it is performing a slow operation.

The default setting for this parameter (10) allows the server to process 10 requests simultaneously. This value may be changed to adjust the tradeoff between concurrency and memory requirements.

## Cache Interrupt Parameters

You can set the cache-interrupt parameters to cause the server to automatically discard cached information about a directory or file when an OpenVMS user tries to access the directory or file on the disk. The `USE-DIRECTORY-BLOCKING-ASTS` and `USE-FILE-BLOCKING-ASTS` parameters control whether the server flushes the cache in this situation.

These parameters can be set to 1 (on) or 0 (off). By default, they are both on, causing the server to discard the cached file header and all data buffers for a directory or file whenever an OpenVMS user attempts to access it on disk. These parameters must be set to 1 (one) to allow PC clients to use the PC-NFSD remote printing function. A setting of 1 also ensures that client users almost always receive the directory or file as it exists on disk. This consistency comes at the expense of the overhead of the additional interrupts and disk reads.

## Cache-Timing Parameters

Because cached information may not be automatically updated if the directory or file is changed on the disk, the server periodically discards cached information. This requires a reread from disk the next time the information is needed.

The cache-timing parameters control the intervals at which channels are deassigned, and at which cached headers and cached data buffers are discarded. One of the parameters controls the polling interval at which the other parameters are checked.

Several of the cache-timing parameters distinguish between idle and active channels and cached data. An idle entity is one that is not being accessed by any client; an active entity is one that is in use.

Some of the cache timing parameters apply only to directories or only to files. Directory settings affect the speed at which local OpenVMS users see files created and deleted; file settings affect the speed at which users see file contents created and deleted.

## Cache Maintenance Interval Parameters

The `FILE-CACHE-TIMER-INTERVAL` parameter determines how often the NFS server scans the cache, polls the other parameters to see if their timers have expired, and processes those that have.

The default setting for the `FILE-CACHE-TIMER-INTERVAL` parameter (30 seconds) is normally not changed during configuration.

## Channel Deassignment Parameters

The `READ-ONLY-FLUSH-AGE` and `READ-WRITE-FLUSH-AGE` parameters determine how long idle channels can remain assigned to a file.

The `READ-ONLY-FLUSH-AGE` parameter applies to files that have been opened for read operations only; the `READ-WRITE-FLUSH-AGE` parameter applies to files that have been opened for both read and write operations. Closing a channel does not discard the data in the file headers and data buffers; clients can continue to access the cached data without requiring that the file be reopened.

The default values are 180 seconds for read-only channels and 60 seconds for read-write channels. You can shorten or lengthen the timer intervals to adjust tradeoffs between improved response time and the overhead of keeping channels assigned.

## Cache Refresh Parameters

The `DIRECTORY-INFO-IDLE-FLUSH-AGE`, `DIRECTORY-INFO-FLUSH-AGE`, `FILE-INFO-IDLE-FLUSH-AGE`, and `FILE-INFO-FLUSH-AGE` parameters control how long cached headers and data buffers for a directory or file can remain in the cache.

As previously indicated, unless the cache-interrupt parameters are on, cached headers and buffers are not automatically discarded whenever an OpenVMS user attempts to access directories and files on disk. The cache-flush parameters specify a period after which the server discards cached information (requiring rereads from disk if the information is needed again). Two of the parameters control idle intervals, and two control active intervals.

Each setting for the four parameters represents a tradeoff between response time and concurrency between information stored in the cache and on the disk.

The default setting for the `DIRECTORY-INFO-IDLE-FLUSH-AGE` parameter is 150 seconds. The default setting for the `DIRECTORY-INFO-FLUSH-AGE` is 300 seconds. (In combination, these settings specify that cached directory information is discarded after 300 seconds if the information is in use, but discarded after 150 seconds if the information is not in use.)

The default setting for the `FILE-INFO-IDLE-FLUSH-AGE` parameter is 600 seconds. The default setting for the `FILE-INFO-FLUSH-AGE` parameter is 1200 seconds.

You can raise or lower any of the default settings, but do not set either directory parameter below 15 seconds or the server will be unable to complete directory operations.

## Cache Size Parameters

The cache-size parameters for the directory and file cache determine the maximum numbers of channels, file headers, and data buffers that can simultaneously exist for the cache or for a given file system. The settings for each of these parameters reflect tradeoffs between response time and memory requirements.

In addition, in combination with `SYSGEN`'s `VIRTUALPAGECNT` parameter and allowable page blocks, three of these parameters affect the maximum amount of memory that the variable portion of the cache can use at a time. As described in the *NFS Server Memory Considerations* section, those parameters are `MAXIMUM-CACHE-FILES`, `MAXIMUM-CACHE-BUFFERS`, and `MAXIMUM-OPEN-CHANNELS`.

## OpenVMS Channel Usage Parameters

The `MAXIMUM-OPEN-CHANNELS` and `MAXIMUM-FILESYSTEM-CHANNELS` parameters determine the maximum number of open channels allowed simultaneously for the cache as a whole and for single file systems on a per-mount-point basis.

When a set limit is reached, and a request to access a new directory or file is received, the server deassigns the oldest open channel and uses the channel to complete the new request, ignoring the setting of the `READ-ONLY-FLUSH-AGE` or `READ-WRITE-FLUSH-AGE` parameter.

The default setting for both parameters is 50. You can change either value to adjust the tradeoff between response time and memory requirements. Do not increase them to greater than 90% of the value for the OpenVMS `SYSGEN` parameter `CHANNELCNT`, which determines the maximum number of channels available to the OpenVMS system as a whole. The 10% buffer between the values is required to handle the OpenVMS channels used for operations other than server operations, and to handle the times when the server is briefly allowed to exceed the `MAXIMUM-OPEN-CHANNELS` value (for example, the period between the time when the server opens a channel that causes the limit to be exceeded and the time when it closes another channel to observe the limit).

If the server runs out of OpenVMS channels, an `ENOBUFFS` error is returned to the client that requested the additional channel (see the *MultiNet Messages, Logicals, and DECnet Applications* book).

## Cache Memory Requirements Parameters

The `MAXIMUM-CACHE-FILES`, `MAXIMUM-FILESYSTEM-FILES`, `MAXIMUM-CACHE-BUFFERS`, and `MAXIMUM-FILESYSTEM-BUFFERS` parameters determine the maximum number of cached file headers and data buffers allowed simultaneously for the cache as a whole and for single file systems on a per-mount-point basis.

As described earlier, cached file headers contain attribute information for directories and files, and data buffers contain data from directories and files. A cached file header requires about 128 bytes of memory; a data buffer requires about 8 kilobytes.

The default setting for both cached-header parameters is 3000; the default setting for both data-buffer parameters is 500. You can change these settings to adjust the tradeoffs between response time and memory requirements, but increasing any of the values may require increasing the size of the server's virtual address space or page file quota, as described in the *NFS Server Memory Considerations* section.

**Note:** Unless the settings for `MAXIMUM-CACHE-BUFFERS` and `MAXIMUM-FILESYSTEM-BUFFERS` are high enough to allow the cache to hold the largest files the client will access, performance will be severely degraded for those files. Each cached data buffer holds 16 disk blocks.

# Auto Server Cache Sizing

The NFS server limits the number of files it has open at one time based on the BYTLM process quota. For the server to operate properly, it must have sufficient BYTLM to have 30 files open simultaneously. If the server process BYTLM quota is too small to accommodate this requirement, the server issues the following OPCOM messages:

```
%%%%%%%%%% OPCOM 11-APR-2020 14:52:04.87 %%%%%%%%%%%  
Message from user SMITH on NODE1  
NFS Server: Max Accessed files: 11.
```

"11" is the calculated number of simultaneous file accesses the server can have, based on the available BYTLM quota.

If this number is less than 30, the following message also appears:

```
%%%%%%%%%% OPCOM 11-APR-2020 14:52:04.88 %%%%%%%%%%%  
Message from user SMITH on NODE1  
NFS Server: Increase BYTLM to new_value.
```

*new\_value* is the recommended value for the BYTLM process quota.

Use SERVER-CONFIG to set the BYTLM process quota for the NFS server process:

```
$ MULTINET CONFIGURE/SERVER  
SERVER-CONFIG>SELECT NFS  
SERVER-CONFIG>SET PQL-BYTLM nnnnn  
SERVER_CONFIG>EXIT
```

Restart the NFS server to put the new process quota into effect:

```
$ MULTINET NETCONTROL NFS RESTART
```

# Writeback Cache Parameters

The SECONDS-BEFORE-WRITEBACK, MAXIMUM-DIRTY-BUFFERS, and MAXIMUM-WRITE-JOBS parameters control the functions of the optional writeback feature of the directory and file cache.

The directory and files cache normally functions as a write-through cache. In this case, whenever a client is notified that a write request has completed, the data is stored on the disk, and data integrity is guaranteed.

The optional writeback feature greatly increases the speed of write operations (as perceived by the user) by notifying the client that write operations are complete when the data is stored in cache memory on the server, but before it has been written to disk.

This increase in perceived write performance is achieved at the risk of data loss if the OpenVMS server system crashes while a write operation is in progress. During a write operation, data may also be lost if the server encounters an error such as insufficient disk space or disk quota or a hardware write error.

If the server cannot complete a writeback write operation, it discards the write operation, flags the file's cached header to indicate the error, and sends an error message in response to the next request for the file. However, if there is no new request before the affected header is discarded or the next request is from another user, data can be lost.

If you enable the writeback cache feature, you can prevent data losses from occurring during system shutdowns by adding the following line to the server's `SYS$MANAGER:SYSHUTDOWN.COM` file:

```
$ MULTINET NETCONTROL NFS SHUTDOWN
```

**Note:** You can also use this command to perform a simple shutdown of the NFS server.

The writeback cache parameters have the following meanings:

- The `SECONDS-BEFORE-WRITEBACK` parameter determines whether the writeback feature is enabled, and specifies how long the server will delay initiating a write operation after receiving data for a write request. The longer the delay, the greater the chance that the server can coalesce multiple small write operations into fewer, larger, and more efficient operations.

**Note:** This timing parameter is not affected by the `FILE-CACHE-TIMER-INTERVAL` parameter.

The default setting (0) disables the writeback feature. Any other value enables the feature. The recommended value for writeback delay is five seconds; little performance is gained from longer delays.

- If the writeback cache is enabled, the `MAXIMUM-DIRTY-BUFFERS` and `MAXIMUM-WRITE-JOBS` parameters control its operation.
- `MAXIMUM-DIRTY-BUFFERS` sets a limit on the number of buffers that can remain in the cache awaiting writeback before the `SECONDS-BEFORE-WRITEBACK` time has expired. As soon as this limit is reached, the server begins writeback of the oldest buffer. The default setting for this parameter (0) sets no limit.
- `MAXIMUM-WRITE-JOBS` sets a limit on the number of write operations that can be simultaneously processed. When this limit is reached, the server defers starting a new write

operation until a current operation completes. The default setting for this parameter (0) sets no limit.

## Duplicate Request Detection Cache Parameters

The server has a duplicate-request detection cache to store the most recent responses it has sent to clients requesting directory and file access. The `NUMBER-OF-DUPLICATE-REQUESTS-CACHED` parameter defines the number of responses that can be cached.

The duplicate-request detection cache operates in conjunction with the cache the RPC protocol module keeps of the transaction IDs (XIDs) of the last 400 requests it has seen. The RPC layer uses its cache to detect duplicate requests.

For example, if the network layer drops a UDP packet containing a response to a client, the client repeats the request after an interval, and the RPC protocol notifies the NFS server that the request was a duplicate. The server then looks in its duplicate-request detection cache for the response so it can resend it without repeating the original operation.

By default, the cache stores the last 250 responses sent.

**Note:** Too low a value causes the following error message to be printed frequently on the OpenVMS console: "Duplicate Detected but not in cache." Too low a value can also cause an incorrect answer to be sent. A value above 400 has the same effect as 400, which is the maximum number of XIDs stored by the RPC protocol.

## Delete-Behind Cache Parameters

The `MAXIMUM-QUEUED-REMOVES` parameter affects the way client users perceive the speed at which directories and files are deleted.

The OpenVMS file deletion operation is very slow. The NFS server uses its delete-behind queue to hide some of this delay from the client user; when a request to delete a directory or file arrives, the request is answered immediately, but the delete request is usually only queued to the OpenVMS file system.

The `MAXIMUM-QUEUED-REMOVES` parameter limits the number of requests that can be queued; when that number is reached, the next delete request must wait until the next queued request has completed.

This delay can be significant if the next request is to delete a large directory; directory deletions always occur synchronously, and each file in a directory must be deleted before the directory itself is deleted.

Therefore, the parameter's setting defines when, in a series of deletions, the client user perceives the delay in the OpenVMS deletion. The default setting is 25.

## Time Zone Parameters

Although OpenVMS does not track time zones, the NFS server requires this information. The `TIMEZONE` parameter identifies the local time zone for the OpenVMS server. This parameter is also a MultiNet global parameter. If the parameter is set appropriately there, you do not need to set it again as a server global parameter.

The server uses the `TIMEZONE` setting to calculate the offset between Greenwich Mean Time and the local time recorded for directories and files when they are cached and modified in the cache.

Valid `TIMEZONE` settings are the time zone abbreviations; for example, PST (Pacific Standard Time). When the setting defines a U.S. time zone, the server automatically adjusts the time zone to conform to the U.S. Federal Daylight Savings Time rules.

The default setting is GMT (Greenwich Mean Time). If your local time and the time to which your OpenVMS clock is set differ, set the `TIMEZONE` parameter to correspond to the OpenVMS clock.

For more information about MultiNet's handling of time, see Chapter 14.

## Special Debugging Parameters

The following special debugging parameters exist only to debug the NFS server under unusual circumstances. Do not use them without instructions from Process Software Technical Support.

<code>CRASH-ON-EXCEPTION</code>	<code>MAXIMUM_DEBUG_PRINTS</code>
<code>DEBUG-MESSAGE-CACHE-SIZE</code>	<code>NFSDEBUG</code>
<code>EXIT-ON-EXCEPTION</code>	<code>PRINT-TO-STDOUT</code>
<code>FILECACHE-DEBUG</code>	<code>RPCDEBUG</code>
<code>HIBERNATE-ON-EXCEPTION</code>	



# NFS Troubleshooting Tips

This section describes workarounds for common problems encountered when using the NFS server.

## Approximate Text Size Threshold

If you have many large files, UNIX commands that expect file sizes to be available may take a long time to execute. To set the approximate text size threshold so UNIX commands like `ls -l` execute faster:

1. Set the threshold on the server system.
2. Use the mount option on the client system so the threshold takes effect.

The following example shows how to set the threshold on the server:

```
$ MULTINET CONFIGURE /NFS
NFS-CONFIG>SET APPROXIMATE-TEXT-SIZE-THRESHOLD 250
[Global NFS parameter "approximate-text-size-threshold" set to 250]
NFS-CONFIG>SAVE
[Writing NFS file server config to
MULTINET_ROOT:[MULTINET]NFS.CONFIGURATION]
NFS-CONFIG>RELOAD
Connected to NETCONTROL server on "127.0.0.1"
< Code-Z.EXAMPLE.COM Network Control 5.6 at Mon 18-Aug-2019 9:25PM-PDT
< OK: NFS/RPCLockMgr server configuration reloading
< RPCMOUNT database reloaded
```

To take advantage of the approximate text size threshold, NFS clients must mount the file system with the `/APPROXIMATE_TEXT_SIZE` option. The following example shows how to use the mount option on a UNIX NFS client.

```
% mount -o soft hq:/altusers/alex/test\#/approximate_text
```

## NFS Stream\_LF File Conversion

If you receive an error message related to incompatible file attributes, the following information will help. When you use the `COPY` command to copy a non-Stream\_LF format file to a disk mounted by the NFS client, MultiNet converts the file to Stream\_LF format (by default) to ensure that text files can be shared between OpenVMS and UNIX systems. To preserve the non-Stream\_LF format, use the `/SEMANTICS=NOSTREAM_CONVERSION` qualifier as part of the `NFSMOUNT` command.

For more information on the `NFSMOUNT` command, refer to the command page in the DCL Commands section of the *MultiNet Administrator's Reference*. For more information on NFS default file attributes, see Chapter 28.

# Performance Problems with Large Directories

Because of XQP limitations, you may experience performance problems when processing certain requests on large directory trees. Process Software recommends that you keep fewer than 1,000 files in each directory.

# 25. Configuring the MultiNet NFS 3 Client & Server

## Introduction

MultiNet supports version 3 of the NFS server and client protocols. NFS 3 provides:

- support for 64-bit file sizes and offsets, to handle files larger than 2 gigabytes
- support for asynchronous writes on the server, to improve write performance
- additional file attributes in many replies, to avoid the need to re-fetch them, thus reducing network traffic
- a REaddirPLUS operation, to get file handles and attributes along with file names when scanning a directory
- assorted other improvements

## Server Security & Initial Configuration

Just like the older NFS 2 server, the NFS 3 server provides several features that maintain the integrity of the OpenVMS filesystem.

First, the server requires that the local system must register any user trying to access OpenVMS files. You do this through the PROXY database when you configure the server and through later modifications as needed.

Second, you must export an OpenVMS directory for an NFS user to access it. The server does this through the EXPORT database when you configure the server and through later modifications as needed.

The initial offering of the NFS 3 server utilizes a compiled binary format for its PROXY and EXPORT databases. Initially, these databases will use the existing proxy and export information found in the MultiNet NFS .CONFIGURATION file. To get the data into the proper format for the NFS 3 server, the CONVERT\_NFS tool must be invoked. The CONVERT\_NFS tool will read the NFS .CONFIGURATION

file and set up the v3 only `NFS_EXPORT.DAT` and `NFS_PROXY.DAT` files. The `CONVERT_NFS` tool is invoked from the command line:

```
$ RUN CONVERT_NFS
```

There is minimal output, and two files are created - `NFS_EXPORT.DAT` and `NFS_PROXY.DAT`. At this time, any changes made via `MULTINET CONFIGURE /NFS` will not be picked up by the NFS 3 server until `CONVERT_NFS` is invoked. Future versions of MultiNet will streamline this process.

Refer to the NFS 2 server documentation in the previous chapter for further details on configuring proxies, exports, and the supporting protocols (i.e., portmapper, etc.) The NFS 3 server utilizes the same supporting RPC protocols as the NFS 2 server.

**Note:** The initial implementation of the NFS 3 service (`NFSV3_SERVER`) will run in conjunction with the legacy NFS service (`NFS_SERVER`) or by itself. The only dependency is the `RPCPORTMAP` service. A future version of MultiNet will incorporate an NFS server that supports both NFS v2 and NFS 3 from the same process.

## Client

Either the legacy NFS 2 client or the NFS 3 client can be used, but both cannot be used at the same time. The NFS 3 client supports the mounting of NFS 2 exported directories as well as NFS 3 directories through different ACP images. NFS 3 needs to be specified as part of MultiNet configuration so that the proper driver is loaded.

```
$ MULTINET CONFIGURE/NETWORK
NET-CONFIG> SET NFS3 TRUE
NET-CONFIG> EXIT
```

The existing MultiNet `NFSMOUNT` command will detect if the NFS 3 driver is being used and convert the mount command to use the NFS 3 mount program. Use the information at the end of the chapter to use the NFS 3 mount program directly.

## Mounting Client Directories

NFS 3 clients access OpenVMS files on the NFS server by mounting directories. The `MOUNT v3` protocol services the mount requests from clients attempting to mount an NFS 3 export.

Mounting procedures vary by client and may require superuser privileges, or in the case of PC clients, a username and password. Some clients mount a remote directory automatically when they reboot the system (as in the case of `fstab`). Others mount a remote directory dynamically when they reference the remote file (as with an `automount`).

Mount procedures require the following information:

- The pathname of the exported directory that matches the pathname in the `EXPORT` database
- The name of the host running the server that contains the files you want mounted
- A pathname on the client designated as the mount point

Below is a mount command provided by a UNIX client to the `NFSV3_SERVER` running on host `IRIS`, using the defined export of `NFS0 : [USER.MNT]` . The export is mounted onto the local `/mnt` partition.

```
# mount IRIS:DKA0:\[USERS.MNT\] /mnt
```

In the example, `IRIS` is the name of the MultiNet host. `DKA0 : [USERS .MNT]` is the exported directory. `/mnt` is the mount point on the UNIX client host.

Check your NFS client documentation before mounting directories.

## File Formats

The NFS protocol does not define standard file and record formats or a way of representing different types, such as text or data files. Each operating system can have a unique file structure and record format.

The server provides access to all OpenVMS files. However, even though an NFS client can access a file, the client may not be able to correctly interpret the contents of a file because of the differences in record formats.

The UNIX operating system stores a file as a stream of bytes and uses a line feed (LF) character to mark the end of a text file line. PC systems also store a file as a stream of bytes, but use a carriage-return/line-feed (CRLF) character sequence to mark the end of a text file line. PC systems sometimes also use a Ctrl/Z character to mark the end of a file.

The OpenVMS operating system, with its Record Management Services (RMS), provides many file organizations and record formats. RMS supports sequential, relative, and indexed file organizations. It also supports `FIXED`, `STREAM`, `STREAM_CR`, `STREAM_LF`, `UNDEFINED`, `VARIABLE`, and variable with fixed size control area (VFC) files.

NFS clients most commonly need to share text files. `STREAM` is the RMS record format that most closely matches PC text files. `STREAM_LF` is the RMS record format that most closely matches UNIX text files.

In OpenVMS, you can store standard text files in `VARIABLE`, `STREAM_LF`, or `VFC` record format. Most OpenVMS utilities can process these text files regardless of the record format because the utilities access them through RMS.

The intent of the server is to provide convenient access to the majority of OpenVMS files. Because many OpenVMS text files are `VARIABLE` or `VFC` format, the server converts these files to `STREAM` or `STREAM_LF` format as it reads them.

## Reading Files

The server reads all files (except `VARIABLE` and `VFC`) block by block without interpreting or converting them. It reads `VARIABLE` and `VFC` files by converting them to `STREAM` or `STREAM_LF`, based on a selected option. The file on the NFS server remains unchanged.

The server's automatic file conversion process can cause a slow reading of `VARIABLE` and `VFC` files. For example, in returning the file size, it reads the entire file. Full directory listings can also be slow if the directory contains a number of `VARIABLE` or `VFC` record format files. If you need frequent access to these files, consider converting them using the OpenVMS `CONVERT` utilities.

## Writing Files

By default, the server creates `STREAM_LF` files, but can also create `STREAM` files on demand. It writes all files except `VARIABLE` and `VFC` block by block without interpreting or converting them. If an NFS client tries to write to or change the size of an existing file not having `STREAM`, `STREAM_LF`, `STREAM_CR`, `FIXED`, or `UNDEFINED` format, the server returns an `EINVAL` error.

## Implementation

This section describes the server restrictions and implementation of the Network File System (NFS) protocol. The material presented here requires a thorough understanding of the protocols. It does not explain or describe the protocols.

# Restrictions

The server has the following OpenVMS-related restrictions:

- The server supports Files-11 ODS-2 structure level disks, ODS-5 formatted disks, and any CD-ROM format.
- The server does not implement volume protection. All exported devices should be public devices.
- The server does not generate security or audit alarms. However, the server writes access violations to log file `MULTINET:NFSV3_SERVER.LOG`.
- When creating files and directories, the server sets the owner UIC of the file or directory to the UIC derived from the UID/GID in the create request authentication information or to the UID/GID in the set attributes information (if available).

# NFS Protocol Procedures

The server implements the following NFS protocol (version 3) procedures (while continuing to support version 2):

Procedures	Description
ACCESS (v3 only)  (access)	The server determines the access rights that a user, as identified by the credentials in the request, has with respect to a file system object.
COMMIT CACHED WRITE DATA  (v3 only)  (commit)	The server forces data to stable storage that was previously written with an asynchronous write call.
CREATE FILE  (create)	The server creates files using the record format specified in the <code>EXPORT</code> database entry. The client may specify one of 3 methods to create the file:

	<p>UNCHECKED: File is created without checking for the existence of a duplicate file.</p> <p>GUARDED: Checks for the presence of a duplicate and fails the request if a duplicate exists.</p> <p>EXCLUSIVE: Follows exclusive creation semantics, using a verifier to ensure exclusive creation of the target.</p>
<p>GET ATTRIBUTES</p> <p>(getattr)</p>	<p>Gets a file's attributes. The server handles certain file attributes in ways that are compatible with the OpenVMS system. These attributes are:</p> <p><b>File protection</b> - The server maps the OpenVMS file protection mask to the UNIX file protection mask.</p> <p><b>Number of links</b> - Although OpenVMS supports hard links, it does not maintain a reference count. Therefore, the server sets this value to 1 for regular files and 2 for directory files.</p> <p><b>UID/GID</b> - The server maps a file owner's UIC to a UID/GID pair through the PROXY database.</p> <p><b>Device number</b> - The server returns the device number as -1.</p> <p><b>Bytes used</b> - The total number of bytes used by the file.</p> <p><b>Filesystem id</b> - The server returns the filesystem ID as 0.</p> <p><b>Access, modify, status change times</b> - The OpenVMS system does not maintain the same file times as NFS requires. The server returns the OpenVMS revision (modify) time for all three NFS times.</p> <p>For directory files, the server returns the access, status change, and modify times as a reasonably recent time, based on the time of the last server-initiated directory change, and the NFS_DIRTIME_TIMER parameter. This is a benefit to clients that cache directory entries based on the directory times.</p>



	<p>OpenVMS bases its time on local time, while UNIX bases its time on Universal time (or Greenwich mean time), and these times may not agree. The offset from Universal time specified when configuring MultiNet resolves the difference between local and Universal time.</p>
<p>GET DYNAMIC FILESYSTEM INFO (v3 only)</p> <p>(fsstat)</p>	<p>The server provides volatile information about a filesystem, including:</p> <ul style="list-style-type: none"> <li>• total size and free space (in bytes)</li> <li>• total number of files and free slots</li> <li>• estimate of time between file system modifications</li> </ul>
<p>GET FILESYSTEM STATISTICS (v2 only)</p> <p>(statfs)</p>	<p>Returns filesystem statistics. The server handles certain file attributes in ways that are compatible with the OpenVMS system. These attributes are:</p> <p><b>Block size</b> - The block size is 1024.</p> <p><b>Total number of blocks</b> - The total number of blocks is the <code>SYS\$GETDVI MAXBLOCK</code> parameter divided by 2.</p> <p><b>Blocks free</b> - The number of blocks free is the <code>SYS\$GETDVI FREEBLOCK</code> parameter divided by 2.</p> <p><b>Blocks available</b> - The number of blocks available to unprivileged users is the same as the number of blocks free.</p>
<p>GET STATIC FILESYSTEM INFO (v3 only)</p> <p>(fsinfo)</p>	<p>The server provides nonvolatile information about a filesystem, including:</p> <ul style="list-style-type: none"> <li>• preferred and maximum read transfer sizes</li> <li>• preferred and maximum write transfer sizes</li> <li>• flags for support of hard links and symbolic links</li> <li>• preferred transfer size of <code>readdir</code> replies</li> <li>• server time granularity</li> <li>• whether or not times can be set in a <code>setattr</code> request</li> </ul>
<p>LINK</p> <p>(link)</p>	<p>Creates a hard link to a file. The server stores the link count in an application access control entry (ACE) on the file.</p>

<b>LOOKUP FILE</b>  (lookup)	Looks up a file name. If the file name does not have a file extension, the server first searches for a directory with the specified name. If the server fails to locate a directory, it searches for the file name without an extension.
<b>MAKE DIRECTORY</b>  (mkdir)	Creates a directory. The OpenVMS system does not allow the remote host to create more than eight directory levels from the root of the OpenVMS filesystem. The server ignores access and modifies times in the request.
<b>READ DIRECTORY</b>  (readdir)	Reads a directory. The server returns file names using the filename mapping scheme as specified in the EXPORT database entry. The server also drops the VMS version number from the file name for the highest version of the file.
<b>READ DIRECTORY PLUS ATTRIBUTES</b> (v3 only)  (readdirplus)	In addition to file names, the server returns file handles and attributes in an extended directory list.
<b>READ FROM FILE</b> (read)	Reads from a file. The server converts VARIABLE and VFC files to STREAM or STREAM_LF format (depending on the option set) as it reads them. The server returns EOF when detected.
<b>REMOVE DIRECTORY</b>  (rmdir)	Deletes a directory.
<b>REMOVE FILE</b>  (remove)	Deletes a file.

<p>RENAME FILE</p> <p>(rename)</p>	<p>Renames a file. If the destination filename is the same as an existing filename and the destination filename does not have a zero or negative version number, the server overwrites the existing file.</p>
<p>READ LINK</p> <p>(readlink)</p>	<p>Reads the contents of a symbolic link.</p>
<p>SET ATTRIBUTES</p> <p>(setattr)</p>	<p>Sets file attributes. The server handles certain file attributes in ways that are compatible with the OpenVMS system. These attributes are:</p> <p><b>File protection</b> - The server maps the UNIX file protection mask to the OpenVMS file protection mask, as shown earlier in this chapter.</p> <p><b>UID/GID</b> - The client changes the file owner's UIC. The PROXY database maps the new UID/GID to an OpenVMS UIC. If the server cannot locate the new UID/GID in the database, it returns an error and does not change the owner UIC.</p> <p><b>Size</b> - If the file size is larger than the allocated size, the server extends the file. If the size is 0, the server truncates the file and sets the record attributes to sequential STREAM_LF. You cannot change the size of variable length or VFC files (except to zero).</p> <p><b>Access time</b> - Changing the access time has no effect on the OpenVMS system.</p> <p><b>Modify time</b> - The modify time updates the OpenVMS revision time.</p>
<p>SYMBOLIC LINK</p> <p>(symlink)</p>	<p>Creates a symbolic link. The server creates the file with an undefined record structure and uses an application ACE on the file to mask it as a symbolic link.</p>
<p>WRITE TO FILE</p> <p>(write)</p>	<p>Writes to a file. The server does not allow a remote host to write to a directory file, or to VARIABLE and VFC files.</p>

If the server allowed a remote host to write to an existing OpenVMS file that was not a `STREAM_LF` or fixed-length record format file, the file could become corrupted. The server does not allow a remote host to explicitly change the record format of an OpenVMS file.

The server can return the non-standard NFS error `ETXTBSY` (26) and `EINVAL` (22). The server returns `ETXTBSY` when an OpenVMS user has a file open for exclusive access and an NFS user tries to use the file in a way that is inconsistent with the way the OpenVMS user opened the file. The server returns `EINVAL` if an NFS user tries to write to or change the size of a `VARIABLE` or `VFC` record format file.

With Version 3, the server supports asynchronous writes (see `commit`).

## Troubleshooting

If you are experiencing network communication-related problems on the NFS 3 server, please check the following items:

1. Make sure MultiNet is running on the OpenVMS system.
2. Confirm the RPC service is running with the following command at the DCL prompt:

```
$ MULTINET SHOW /RPC
```

3. Make sure the server is running. If not, start it by entering the following command at the DCL prompt:

```
$ MULTINET NETCONTROL NFSV3 RESTART
```

4. To verify general connectivity between the two systems, try using FTP or TELNET. For example, try to open a TELNET connection with the remote host in question. If another product is not available on your system, try using the `PING` utility.
5. Verify the internet addresses the local host and the remote hosts are using. If your local network includes a gateway, also verify the gateway address.

6. If you are experiencing problems performing NFS operations from a NFS client, check the server's `NFSV3_SERVER.LOG` file. It may contain messages that can help isolate the problem.

## Managing an Existing NFSv3 Configuration

MultiNet includes `MANAGE_NFS3` to manage NFS 3 client and server configurations.

```
$ run multinet:manage nfs3
```

Changes made to the configuration with this program are NOT made to the NFS 2 database. The `CONVERT_NFS3` program will overwrite the NFS 3 databases. The `MANAGE_NFS3` program has the following commands to manage the configuration:

---

# ADD EXPORT

*NFS server only.*

Adds an entry to the EXPORT database that lets the NFS server export the server filesystems to a remote NFS client. Users at the NFS-Client can then mount the server filesystems. Requires write access to the MULTINET:NFS\_EXPORT.DAT file. The EXPORT database is dynamic. Entries you add to the database become valid immediately. You do not need to restart the server.

If you are adding entries to the EXPORT database for the first time, read the *EXPORT Database* section in Chapter 24 of the *MultiNet Installation and Administration Guide*.

## Format

```
ADD EXPORT "nfs-path" vms-directory
```

## Parameters

### *nfs-path*

NFS-style pathname used to reference the exported directory. Typically expressed as a UNIX-style pathname. Enclose in quotation marks (" ").

Although *nfs-path* can be arbitrary, it usually reflects the actual OpenVMS directory path. The NFS client user must refer to the same *nfs-path* in naming the mount point.

### *vms-directory*

Directory on the local OpenVMS server that you want to export. The directory must include the device specification, as in the following example:

```
$DISK1:[SALES.RECORDS]
```

When you export a directory, the NFS client user can potentially have access to all files and directories below the export point. The device you export should be a "public" device. The server does not implement volume protection. Also, the server supports Files-11 ODS-2 and ODS-5 structure level disks.

## Qualifiers



**Note:** Many of the following qualifiers are specific to applications running on certain hosts. In these cases, it is critical to use the /HOST qualifier in combination with these qualifiers.

**/HOST=(*host*[ , *host* . . . ])**

Only specified host(s) can have access to the exported OpenVMS directory. `MANAGE_NFS3` allows either host names or internet addresses. Use the parentheses only if you specify a list of hosts (separated by commas). If you omit /HOST, any host can mount the exported directory.

**/CONVERT={*STREAM\_LF* | *STREAM\_CRLF*}**

**/NOCONVERT**

/CONVERT converts files on reads to either `STREAM_LF` (the default) for UNIX systems or `STREAM_CRLF` for PC systems. /NOCONVERT disables this conversion and must be specified when using the server together with MultiNet's NFS client.

**/EXPLICIT\_MOUNT**

**/NOEXPLICIT\_MOUNT**

/EXPLICIT\_MOUNT prevents users from subsequently mounting subdirectories of the mount point. /NOEXPLICIT\_MOUNT, the default, allows subdirectory mounts.

**/FILENAME={ *SRI* | *ODS5* | *PATHWORKS* | *PATHWORKS\_CASE* }**

Uses the SRI International, ODS5, or PATHWORKS filename mapping schemes.

SRI is the default scheme between UNIX and OpenVMS systems.

ODS5 uses minimal mapping to get around ODS-5 file naming restrictions. If the disk or system doesn't support ODS-5, it falls through to SRI.

PATHWORKS specifies non-case-sensitive filename mapping.

PATHWORKS\_CASE specifies case-sensitive filename mapping.

**/HIGHEST\_VERSION**

**/NOHIGHEST\_VERSION**

/HIGHEST\_VERSION returns only the highest version of files in directory requests.

/NOHIGHEST\_VERSION, the default, does not. All file versions still exist in either case.

**/PRIVILEGED\_PORT**

**/NOPRIVILEGED\_PORT**

/PRIVILEGED\_PORT requests that incoming requests originate from privileged ports only.

/NOPRIVILEGED\_PORT, the default, does not.

**/PROXY\_CHECK**

**/NOPROXY\_CHECK**

/PROXY\_CHECK specifies that mount requests only originate from users having mappings in the

PROXY database. /NOPROXY\_CHECK, the default, does not.

**/RFM=*option***

Record format (RFM) of newly created files. The options are STREAMLF, STREAMCR, STREAM,

FIXED, and UNDEFINED.

**/SERVER\_ACCESS**

**/NOSERVER\_ACCESS**

/SERVER\_ACCESS requests the server to do access checking. /NOSERVER\_ACCESS, the default,

requests that both the server and client do the checking.

**/SUPERUSER\_MOUNT**

**/NOSUPERUSER\_MOUNT**

/SUPERUSER\_MOUNT requests that only the superuser can mount a file system.

/NOSUPERUSER\_MOUNT, the default, does not.

**/VERSION={ DOT | SEMICOLON (default) | ALL | HIGHEST }**

DOT changes the file version display for exported filesystems to `file.ext.version` (a dot) for UNIX compatibility instead of the usual `file.extension;version` (a semicolon).

SEMICOLON (default) uses the regular semicolon.

ALL exports files with version numbers intact rather than the default of leaving the highest numbered version unnumbered.

HIGHEST is a synonym for /HIGHEST\_VERSION. Do not use DOT with SEMICOLON.

**/WRITE**

**/NOWRITE**



`/WRITE` requests that the client have read-write access to the filesystem. `/NOWRITE`, the default, requests that the client have read access only.

## Example

Exports the directory `SALES.RECORDS` on device `$DISK1`: as path `/vax/records` to hosts `ORCHID` and `ROSE`. Any subdirectories below `SALES.RECORDS` are also accessible. However, hosts `ORCHID` and `ROSE` cannot have access to or mount directories above `SALES.RECORDS` or other `SALES` subdirectories.

```
$ ADD EXPORT "/vax/records" $DISK1:[SALES.RECORDS] /HOST=(ORCHID,ROSE)
```

---

# ADD GROUP

*NFS client only.*

Adds an entry to the GROUP database that associates an OpenVMS user with an NFS group or list of groups. Requires SYSPRV privilege and write access to the MULTINET:GROUP.DAT file.

If the GROUP database does not exist, use the CREATE GROUP command first to create an empty one. Use the REMOVE GROUP command to remove a group from the database.

**Note:** The GROUP database is static. Use the RELOAD command when you modify it.

## Format

```
ADD GROUP nfs-group vms-identifier
```

## Parameters

### *nfs-group*

NFS group number found in the /etc/group file on the server. For example, if the users group appears in the /etc/group file as:

```
users:x:15:  
use 15 as the nfs-group.
```

### *vms-identifier*

Associates either an OpenVMS rights identifier or UIC (or wildcarded UIC) with the NFS group. Only associate one *vms-identifier* per NFS group. Use either of the following formats to enter the value:

Format	Description
Name	OpenVMS rights identifier or username

Value	UIC value in [group,member] or %Xnnnnnnnn format; you can use wildcard entries such as [200,*].
-------	---

"Name" and "value" correspond to the columns associated with entries in the OpenVMS rights database. To have access to this database, use the commands:

```
$ SET DEFAULT SYS$SYSTEM
$ RUN AUTHORIZE
UAF>SHOW/IDENTIFIER *
```

For example, the following line may appear in the rights database:

Name	Value	Attributes
----	----	-----
USER	[000200,000200]	

## Qualifier

**/HOST=(host[,host...])**

Server host(s) on which the group identification is valid. If omitted, any remote host is valid for the group. /HOST accepts either host names or internet addresses. Use the parentheses with multiple host entries.

## Examples

1. Associates NFS group number 15 on server host IRIS with the "value" [200,\*], meaning "any user in group 200."

```
$ ADD GROUP /HOST=IRIS
  Group: 15
  Identifier: [200,*]
```

The *nfs-group* number derives from the entry in the */etc/group* file on the server for the users group:

```
>cat /etc/group
staff:*:10:
users:*:15:
```

2. Associates NFS group number 15 with the OpenVMS rights identifier, USERS. As in Example 1, the *nfs-group* number derives from the entry in the */etc/group* file on the server. Assuming that the USERS rights identifier exists in the rights database, any user granted this identifier would be in the group corresponding to GID 15 in NFS.

```
$ ADD GROUP 15 USERS
```

The resulting ADD GROUP entry would appear in the GROUP database as follows:

```
NFS GROUP Database
Group  Name      Value          Host(s)
-----  ----      -
15     USERS      %X8001000C
```

---

# ADD PROXY

*NFS client and NFS server.*

Registers an NFS or remote user as an OpenVMS username in the PROXY database. Requires SYSPRV privilege and write access to the MULTINET:NFS\_PROXY.DAT file.

**Note:** If you omit the /CLIENT or /SERVER qualifier, or do not define the MULTINET\_NFS\_DYNAMIC\_PROXY logical accordingly, you must use the RELOAD PROXY command to reload the database. (For details, see the RELOAD in this chapter.)

## Format

ADD PROXY *vms-username*

## Parameter

***vms-username***

OpenVMS username to which you want to map an NFS user ID. The username must appear as in the OpenVMS User Access File (SYSUAF.DAT).

## Qualifiers

The /HOST, /UID, /GID, or /NFS qualifiers make the PROXY entry more restrictive. When you omit a qualifier, NFS-OpenVMS interprets it as a wildcard. For example, the command ADD PROXY SMITH/UID=210 creates an entry that lets a user with UID=210, but with any GID and from any host, use OpenVMS username SMITH.

**/HOST=(*host* [, *host* . . . ])**

Host(s) from which the UID/GID identification is valid. Specify at least one host name. If omitted, MANAGE\_NFS3 allows any remote host with the matching identification. /HOST accepts either host names or internet addresses. Use parentheses for multiple hosts.

**/UID=*uid***

User's ID (UID). If omitted, `MANAGE_NFS3` accepts any UID for the `vms-username`.

**/GID=*gid***

User's group ID (GID). If omitted, `MANAGE_NFS3` accepts any GID for the `vms-username`.

**/CLIENT**

**/NOCLIENT**

`/CLIENT` notifies the client to immediately update its loaded PROXY database with an entry for `vms-username`. `/NOCLIENT`, the default, does not notify the client. This overrides any default action specified using the `MULTINET_NFS_DYNAMIC_PROXY` logical.

**/SERVER**

**/NOSERVER**

`/SERVER` notifies the server to immediately update its loaded PROXY database with an entry for `vms-username`. `/NOSERVER`, the default, does not notify the server. This overrides any default action specified using the `MULTINET_NFS_DYNAMIC_PROXY` logical.

## Examples

The following examples range from most restrictive to least restrictive:

1. Registers a user with `UID=210` and `GID=5` at host `ROSE` to OpenVMS username `SMITH` for the NFS server only.

```
$ ADD PROXY SMITH /UID=210 /GID=5 /HOST=ROSE /SERVER
```

2. Registers a user with `UID=210` and `GID=5` to OpenVMS username `SMITH` and dynamically reloads the PROXY database on both the client and server.

```
$ ADD PROXY SMITH /UID=210 /GID=5 /CLIENT /SERVER
```

3. Registers any user with `GID=5`, any UID, and at any host to OpenVMS username `JONES`.

```
$ ADD PROXY JONES /GID=5
```

4. Registers any user from host `ORCHID` to OpenVMS username `JONES`.

```
$ ADD PROXY JONES /HOST=ORCHID
```

---



# CREATE EXPORT

*NFS server only.*

Creates an empty EXPORT database. Requires write access to the MULTINET:NFS\_EXPORT.DAT file.

**Note:** NFS server installations create an empty EXPORT database. Use this command to supersede an existing EXPORT database only.

## Format

```
CREATE EXPORT
```

## Example

Shows the current EXPORT database, overwrites it, and shows that the database is now empty.

### SHOW EXPORT

```
NFS EXPORT Database
```

Path	Directory	Host(s)
----	-----	-----
/usr	\$DISK1:[SALES.RECORDS]	SIGMA

### CREATE EXPORT

### SHOW EXPORT

```
no EXPORT entries found
```

---



# CREATE GROUP

*NFS client only.*

Creates an empty GROUP database. Requires write access to the MULTINET:NFS\_GROUP.DAT file.

**Note:** Client installation creates an empty GROUP database. Only use this command to supersede an existing GROUP database.

## Format

```
CREATE GROUP
```

## Example

Shows the current GROUP database, overwrites it, and shows that the database is now empty.

```
SHOW GROUP
NFS GROUP Database
Group      Name      Value      Host(s)
-----
15         GROUP    %X8001000B
15         GROUP_16 %X8001000E
CREATE GROUP
SHOW GROUP
    No entries in GROUP database
```

# CREATE PROXY

*NFS client and server.*

Creates an empty PROXY database. Requires write access to the `MULTINET:NFS_PROXY.DAT` file.

**Note:** Client and server installation creates an empty PROXY database. Only use this command to supersede an existing PROXY database.

## Format

```
CREATE PROXY
```

## Example

Shows the current PROXY database, overwrites it, and shows that the database is now empty.

```
SHOW PROXY  
NFS_PROXY Database  
Username      UID      GID      Host(s)  
-----      ---      ---      -----  
BART          1116     15  
MARGE         1115     15  
LISA          1117     16  
HOMER         -2       -2
```

```
CREATE PROXY  
SHOW PROXY  
no PROXY entries found
```

---

# FIND PROXY

*NFS client and server.*

Locates and displays a single entry in the PROXY database. Requires read access to the MULTINET:NFS\_PROXY.DAT file.

On the client, use this command to find the UIC assigned a specific user.

On the server, use this command to determine which OpenVMS username the server uses when it receives a request from the specified UID, GID, and host name.

## Format

FIND PROXY

## Qualifiers

**Note:** You must specify *all three* of the following qualifiers.

**/HOST=host-name**

Host on which the user is valid. This qualifier is required.

**/UID=uid**

User's ID (UID). This qualifier is required.

**/GID=gid**

User's group ID (GID). This qualifier is required.

## Example

Locates an OpenVMS username for an NFS user with UID=210, GID=5, at host ROSE.

```
FIND PROXY /UID=210 /GID=5 /HOST=ROSE
```

```
NFS PROXY Database
```

Username	UID	GID	Host(s)
-----	---	---	-----
SMITH	210	15	ROSE

---

# RELOAD GROUP

*NFS client only.*

Implements changes made to the GROUP database without having to restart the client system. Requires `SYSLCK` privilege.

**Note:** The GROUP database is normally static. The RELOAD command puts the changes into effect. Use this command sparingly. The client can take a significant amount of time to reload the database. The reloading process blocks NFS activity.

## Format

RELOAD GROUP

---

# RELOAD PROXY

*NFS client and server.*

Implements changes made to the PROXY database without having to restart the client or server. Not necessary if the MULTINET\_NFS\_DYNAMIC\_PROXY logical was defined as CLIENT or SERVER or the combination of the two. Requires SYSLCK privilege. When both CLIENT and SERVER are specified the logical should be defined as if it is a search list, not a single value.

**Note:** The PROXY database is normally static. The RELOAD PROXY command puts the changes into effect. Use this command sparingly. The client can take a significant amount of time to reload the database. The reloading process blocks NFS activity.

## Format

```
RELOAD PROXY [vms-username [, vms-username, ...]
```

## Parameter

***vms-username***

Reloads only the PROXY database entries for the specified username (or list of usernames separated by commas). This is useful for notifying the client or server of changes to the OpenVMS SYSUAF.DAT file, such as changes to the rights list or user privileges.

## Qualifiers

**Note:** If you omit both qualifiers, the PROXY database reloads on both the client and server.

```
/CLIENT  
/NOCLIENT
```

`/CLIENT` reloads the `PROXY` database on the client only. `/NOCLIENT` does not reload the database on the client.

**`/SERVER`**

**`/NOSERVER`**

`/SERVER` reloads the `PROXY` database on the server only. `/NOSERVER` does not reload the database on the server.

---

# REMOVE EXPORT

*NFS server only.*

Removes an entry from the EXPORT database so that you can remove access to an exported directory for a single host or a list of hosts. Requires write access to the MULTINET:NFS\_EXPORT.DAT file.

**Note:** The EXPORT database is dynamic. Any path that you remove from the database becomes invalid immediately. You do not need to restart the server.

## Format

```
REMOVE EXPORT "nfs-path"
```

## Parameter

*"nfs-path"*

NFS-style pathname used to reference the exported directory. Typically expressed as a UNIX-style pathname. You must enclose the *nfs-path* in quotation marks (" ").

## Qualifier

*/HOST=(host[, host...])*

Removes access to a *nfs-path* for a single host or a list of hosts. If omitted, MANAGE\_NFS3 removes *nfs-path* for all hosts.

## Example

Removes a record from the EXPORT database so that NFS host ORCHID can no longer mount an OpenVMS directory on the */vax/records* pathname.

```
MANAGE_NFS3>REMOVE EXPORT "/vax/records" /HOST=ORCHID
```

---





# REMOVE GROUP

*NFS client only.*

Removes a group mapping from the GROUP database. Requires write access to the MULTINET:NFS\_GROUP.DAT file.

**Note:** The GROUP database is static. The RELOAD command puts changes into effect.

## Format

```
REMOVE GROUP nfs-group [vms-identifier,...]
```

## Parameters

### *nfs-group*

NFS group number. If you specify *nfs-group* alone, MANAGE\_NFS3 removes the entire group from the database.

### *vms-identifier*

OpenVMS rights identifier(s) or UIC(s) associated with the NFS group. If you specify one, MANAGE\_NFS3 removes only that identifier from the database; MANAGE\_NFS3 does not change the remaining entries for that group. See the ADD command for the valid format of *vms-identifier* entries.

## Qualifier

```
/HOST=(server[,server...])
```

Server host(s) on which the group number is valid. Either host names or internet addresses are valid. This qualifier removes the GROUP entry for the specified host(s) only. Use the parentheses with multiple *server* specifications.

## Example

Removes a record from the GROUP database so that you can no longer associate group number 15 with a group account on the client.

```
MANAGE_NFS3>REMOVE GROUP 15
```

---

# REMOVE PROXY

*NFS client and server.*

Removes an entry from the PROXY database. Requires SYSPRV privilege and write access to the MULTINET:NFS\_PROXY.DAT file.

**Note:** If you omit the /CLIENT or /SERVER qualifier, or do not define the MULTINET\_NFS\_DYNAMIC\_PROXY logical accordingly, you must use the RELOAD PROXY command to reload the database. (For details, see the RELOAD command in this chapter.)

## Format

```
REMOVE PROXY vms-username
```

## Parameter

***vms-username***

OpenVMS account you want to remove from the PROXY database. You can use the wildcard \* in place of *vms-username* as long as you use one of the qualifiers below to be more selective about the update.

## Qualifiers

If you omit a /HOST, /GID, or /UID qualifier, the command removes all entries containing the *vms-username* account from the database.

**/HOST=(*server*[ , *server* . . . ])**

Server host(s) on which the user is valid. MANAGE\_NFS3 removes the PROXY entry for the specified host(s) only. Use the parentheses with multiple *server* specifications.

**/GID=*gid***

User's group ID (GID). MANAGE\_NFS3 removes the PROXY entry for the specified GID only.

**/UID=*uid***

User's ID (UID). `MANAGE_NFS3` removes the `PROXY` entry for the specified UID only.

**/CLIENT**

**/NOCLIENT**

`/CLIENT` notifies the client to immediately update its loaded `PROXY` database with an entry for `vms-username`. `/NOCLIENT`, the default, does not notify the client. This overrides any default action specified using the `MULTINET_NFS_DYNAMIC_PROXY` logical.

**/SERVER**

**/NOSERVER**

`/SERVER` notifies the server to immediately update its loaded `PROXY` database with an entry for `vms-username`. `/NOSERVER`, the default, does not notify the server. This overrides any default action specified using the `MULTINET_NFS_DYNAMIC_PROXY` logical.

## Examples

1. Removes authorization for an NFS user at host `MARIGOLD` with `UID=210` and `GID=5` to use the OpenVMS username `SMITH`.

```
MANAGE_NFS3>REMOVE PROXY SMITH /UID=210 /GID=5 /HOST=MARIGOLD
```

2. Removes authorization for all users at host `CROCUS` to use OpenVMS username `JONES`.

```
MANAGE_NFS3>REMOVE PROXY JONES /HOST=CROCUS
```

3. Removes authorization for any user at host `MARIGOLD` to use any OpenVMS username.

```
MANAGE_NFS3>REMOVE PROXY * /HOST=MARIGOLD
```

4. Removes all entries containing the OpenVMS username `SMITH`.

```
MANAGE_NFS3>REMOVE PROXY SMITH
```

5. Removes authorization for a user with UID=210 and GID=5 to use the OpenVMS username SMITH and dynamically reloads the PROXY database on both the client and server.

```
MANAGE NFS3>REMOVE PROXY SMITH /UID=210 /GID=5 /CLIENT /SERVER
```

---

# SHOW GROUP

*NFS client only.*

Displays entries in the client's GROUP database. Requires read access to the MULTINET:NFS\_GROUP.DAT file.

## Format

```
SHOW GROUP [nfs-group]
```

## Parameter

*nfs-group*

NFS group number for which to show database entries. If omitted, MANAGE\_NFS3 displays entries for all groups on the local client.

## Qualifiers

*/HOST=(server[, server...])*

Server host(s) on which the group number is valid. MANAGE\_NFS3 accepts either host names or internet addresses. Use the parentheses with multiple server specifications.

*/OUTPUT=filespec*

Uses the specified file instead of the terminal for output.

## Example

Shows the NFS group number on host IRIS and corresponding OpenVMS group name and value.

```
MANAGE_NFS3>SHOW GROUP /HOST=IRIS
```

```
NFS GROUP Database
```

Group	Name	Value	Host(s)
-----	----	-----	-----
15	USER	[200,*]	IRIS

---





# SHOW MOUNT

*NFS client and server.*

Displays a list of client hosts that mounted a file system served by a specified NFS server. Returns the mounted directories by the pathnames `MANAGE_NFS3` uses to export them, not the directory names as the OpenVMS system knows them.

## Format

```
SHOW MOUNT [server-host]
```

## Parameter

*server-host*

NFS server host from which to get the list of mounted file systems. If omitted, `MANAGE_NFS3` uses the local server.

## Qualifier

*/OUTPUT=filespec*

Uses the specified file instead of the terminal for output.

## Examples

1. Because the user did not specify the server host name, the system displays the full domain name for the local server ZETA. In this example no client hosts have mounted any of the server file system.

```
MANAGE_NFS3>SHOW MOUNT
NFS Mount List

Server: ZETA.example.com
Path      Host
----      ----
```

2. Displays the list of client hosts and directories by pathnames for mounted file systems served by the specified server IRIS.

```
MANAGE_NFS3>SHOW MOUNT IRIS
```

```
NFS Mount List
```

```
Server: IRIS
```

```
Path          Host
```

```
----
```

```
----
```

```
/sales/records    bart.example.com
```

```
/exported/spool   bart.example.com
```

---

# SHOW PROXY

*NFS client and server.*

Displays the contents of the PROXY database. Requires read access to the MULTINET:NFS\_PROXY.DAT file.

## Format

SHOW PROXY [*vms-username*]

## Parameter

***vms-username***

OpenVMS account entries you want to display. If omitted, the system displays the contents of the PROXY database determined by the qualifiers listed below.

## Qualifiers

***/HOST=(server[ , server. . . ])***

Displays the PROXY entries restricted to the specified server host(s) only, or for which there are no host restrictions given. Specify one or more server hosts (if multiple, separate by a comma and use the parentheses).

***/GID=gid***

NFS user's group ID (GID). MANAGE\_NFS3 displays only entries containing the specified GID.

***/UID=uid***

NFS user's ID (UID). The system displays only entries containing the specified UID.

***/OUTPUT=filespec***

Uses the specified file instead of the terminal for output.

## Example

Displays the PROXY database entries for user SMITH.

```
MANAGE_NFS3>SHOW PROXY SMITH  
NFS PROXY Database
```

Username	UID	GID	Host(s)
-----	---	---	-----
SMITH	100	101	

---

# SHOW STATISTICS

*NFS server only.*

Displays statistics information on the NFS server, useful in troubleshooting if problems occur. The server must be running.

## Format

SHOW STATISTICS

## Qualifiers

**/RESET**

Displays the counter information, then resets the counters. Requires OPER privilege.

**/TIMES**

Displays the additional average and maximum times (in milliseconds) for certain NFS requests listed.

**/OUTPUT=*filespec***

Uses the specified file instead of the terminal for output.

## Description

The NFS statistics returned by the command are:

<b>Started</b>	Date and time someone started the server.
<b>Uptime</b>	Total amount of time the server has been running.
<b>Memory in use</b>	Total amount of dynamic memory (in bytes) the NFS server uses. This includes memory allocated for the RPC server routines.
<b>Threads</b>	NFS thread counters give the <i>total</i> threads available, the <i>current</i> number of threads in use, and the <i>maximum</i> number of threads that have been in use at one time.

	<p>These statistics can give an indication of server load. If the <i>maximum</i> number of threads in use at one time is equal to the <i>total</i> threads available, you may want to increase the number of threads defined by the parameter NFS_THREADS.</p>
<b>Files</b>	<p>File system counters include the number of <i>opens</i> and <i>closes</i> performed by the server, the number of files <i>currently open</i>, and the <i>maximum</i> open files at one time since someone started the server.</p> <p>The number of files <i>currently open</i> and the <i>maximum open</i> files at one time can be an indication of the load on the server.</p>
<b>NFS</b>	<p>NFS counters return the total NFS procedure calls, and the total calls for each NFS procedure since you started the server. These counters can give an indication of the load on the server.</p> <p>total is the total number of calls  bad call is the number of bad calls  fail is the number of failed calls  null is the number of null calls  getattr is the number of get attribute calls  setattr is the number of set attribute calls  read is the number of reads  lookup is the number of lookups  mkdir is the number of make directory calls  write is the number of writes  create is the number of creates  remove is the number of removes  rename is the number of renames  rmdir is the number of directory removes  readdir is the number of address reads  statfs is the number of file system statistics calls  link is the number of create link to file calls  symlink is the number of create symbolic link calls  readlink is the number of read from symbolic link calls  other is the number of other calls</p>

<b>RPC</b>	<p>RPC counters provide information on RPC operations. This includes the total number of <i>receives</i>, <i>transmits</i>, <i>XID hits</i>, and <i>duplicate receives</i>.</p> <p>The <i>XID hits</i> counter gives the number of cached replies the NFS server retransmitted. The <i>duplicate receives</i> counter gives the number of times the server received a duplicate request for an operation that was in progress at the time of the request. If either of these counters is excessive you may need to increase the timeout time on the system running the NFS client.</p>
<b>RPC Errors</b>	<p>RPC counters also returns the following error conditions: <i>receive</i> and <i>transmit errors</i>, <i>authentication errors</i>, <i>decode errors</i>, and <i>RPC program errors</i>.</p>
<b>MOUNT</b>	<p>MOUNT counters return the total MOUNT procedure calls, the calls for each MOUNT procedure since someone started the server, the total number of directory mounts since someone started the server, and the number of directories currently mounted.</p> <p>total is the number of MOUNT calls  bad call is the number of bad MOUNT calls  fail is the number of failed MOUNT calls  mount is the number of successful mounts  unmount is the number of successful dismounts  null is the number of null mounts  dump is the number of dumps from MOUNT calls  mnt export is the number of exported mounts  cur mount is the number of current mounts</p>

## Example

The command description section describes the output parameters for this example. The `/TIME` qualifier includes the average and maximum times for the indicated NFS requests.

```
MANAGE NFS3>SHOW STATISTICS /TIME
NFS Server Statistics
Started: 1-FEB-2014 07:24:05 Uptime: 14 07:05:53 Memory in use: 1414850
Threads:      total      40 current    0 max        11
Files:       opens       54 closes    54 cur. open  0 max.open   5
NFS:        total      2519 bad call   0 fail       0
  null      6 getattr   149 setattr   6 read       396 lookup   1381
```

```

ave:      0 ms          7 ms          82 ms          78 ms          20ms
max:      0 ms          40 ms         100 ms         180 ms          50 ms
  mkdir   0 write      396 create      6 remove       12 rename       18
ave:      0 ms          38 ms          83 ms          38 ms          117 ms
max:      0 ms          510 ms         90 ms          120 ms          130 ms
  rmdir   0 readdir    51 statfs       1 link          0 symlink       0
ave:      0 ms          32 ms          10 ms          0 ms            0 ms
max:      0 ms          230 ms         10 ms          0 ms            0 ms
  readlink 0 other      0 adfread      97 adfwrite     6
ave:      0 ms          0 ms           7 ms           27 ms
max:      0 ms          0 ms           50 ms          30 ms

RPC:
  recv    2520 xmit    2520 xid hits  0 dup recv  0
RPC errors:
  recv    0 xmit    0
  authweak 0 authother 0 decode    0 noproc    0 noprog    0
  progvrs  2 systemerr 0

MOUNT:
  total    1 bad call  0 fail      0
mount     1 unmount  0 null      0 dump      0 mnt export 0
mounts    1 cur. mount 1

```

---



# UNMOUNT ALL

*NFS client only.*

Removes all the mount list entries for the local client host on the specified NFS server or servers. Useful for notifying the remote server host that the server file systems are no longer mounted on the client in the event that the client system goes down and you need to reboot it.

**Note:** Unmounting is not the same as dismounting. `UNMOUNT ALL` does not dismount a mounted file system.

After using `UNMOUNT`, you can use `SHOW MOUNT` (in MultiNet) or `show mount` (on a UNIX system server) to verify that the list entry you requested to be unmounted on the specified server(s) is no longer there. The mount list entries are in the `/etc/rmtab` file on most UNIX systems.

## Format

```
UNMOUNT ALL
```

## Qualifier

```
/HOST=(server, server . . .)
```

Server host or hosts. The parentheses are required for multiple servers. If omitted, the client sends a broadcast message to all local network servers to remove the list entry for the local client host.

## Examples

1. Sends a broadcast message to all local network servers to remove the mount list entry for the local client host.

```
$ UNMOUNT ALL
```

2. Sends a request to hosts TAU and SIGMA to remove the mount list entry for the local client host.

```
$ UNMOUNT ALL /HOST=(TAU, SIGMA)
```

**Note:** The following message can occur after an `UNMOUNT ALL` request sent to a UNIX system server:

```
RPC Client call failed, RPC: Remote system error
```

Ignore this message. However, confirm through a `SHOW MOUNT` command that the mount list entry was, in fact, removed.

---

# Mounting an NFSv3 file system on MultiNet

The MultiNet NFSMOUNT command will translate the existing mount commands and spawn the appropriate command to use the new mount program. It is possible that the command line may become too long, and there are some options that do not translate between the two mount programs so users may want to convert to the new mount program. Define the following symbol:

```
$ CNFSMOUNT := $multinet:cnfsmount
$ CNFSMOUNT server "nfs-path" [mountpoint [logical]]
```

## Parameters

### *server*

Name of the remote server, in domain name or IP address format.

### *"nfs-path"*

Pathname (enclosed in quotation marks) on the remote server. The pathname must match an exported directory, subdirectory, or file of an exported filesystem on the server. (You can use the SHOW EXPORT command in the MANAGE\_NFS3 utility to obtain a list of the exported directories.)

### *mountpoint*

NFS device (and, optionally, directory tree) specification for the local mount point. If specified, this parameter must be in the format:

```
NFSn: [[dir.dir....]] [filename]
```

The value *n* can range from 1 to 9999, and *dir* is a directory level (up to eight in addition to the [000000] directory). If you omit the *mountpoint* specification or specify NFS0:, the client creates an NFSn: [000000] mount point, and increases *n* by one for each subsequent mount.

### *logical*

Optional logical name associated with the volume. The client defines the logical as follows:

- If you mount NFSn: [000000] NFSn:
- If you mount NFSn: [dir.dir] NFSn: [dir.dir.]

The extra dot after the last `dir` in the second definition allows for relative directory specifications. If you perform the following function:

```
$ SET DEFAULT logical: [subdir]
```

the full default definition becomes:

```
NFSn: [dir.dir.subdir]
```

The client places the logical name in the SYSTEM logical name table unless you specify the /GROUP or /SHARE qualifier. The client deletes the logical name from the SYSTEM table when you dismount the volume. The process must have SYSNAM privilege to mount a system mount point. Without SYSNAM or GRPNAM privilege, the user must specify /SHARE for a JOB mount.

## Qualifiers

```
/ACP_PARAMS=( [BUFFER_LIMIT=limit-value]  
              [, DUMP]  
              [, IO_DIRECT=value]  
              [, IO_BUFFERED=value]  
              [, MAX_WORKSET=pages]  
              [, PAGE_FILE=filespec]  
              [, PRIORITY=base-priority]  
              [, WORKSET=pages])
```

Includes SYSGEN ACP and detached process parameters the system manager can set or modify. The SYSGEN parameters that affect ACPs are dynamic. The client applies the ACP parameters only at the initial start of an ACP and ignores them in subsequent mount requests when the client uses the same ACP.

```
/ADF=option
```

```
/NOADF
```

Controls whether you want to use attributes data files (ADFs). These files appear on a non-VMS server as `.$ADF$filename` files and the server uses them to store OpenVMS file attributes. You cannot directly view these files on the client system. The possible ADF `option` values are:

CREATE (the default and forced if /SERVER_TYPE=VMS_SERVER)	If ADFs exist on the server, the client will use, update, and create them for new files.
UPDATE	If ADFs exist on the server, the client will use and update them, but not create them for new files.

USE	If ADFs exist on the server, the client will use them, but not update them nor create them for new files.
-----	---

Avoid using UPDATE and USE. The client may create ADFs anyway in certain cases, such as when renaming files. Also, changing VMS attributes for a hard-linked file may result in inconsistent VMS attributes between the linked files.

**/AUTOMOUNT [= (INACTIVITY=*inactive-time*) ]**

Mounts a server filesystem automatically and transparently when you obtain the pathname. INACTIVITY specifies a maximum inactive period for the mount attempt. When the client reaches this period, it unmounts the pathname. Specify the time in delta (see *Delta Time Examples*). The default is five minutes (:5). Seconds are rounded to the nearest minute.

**/BACKGROUND [= (DELAY=*delay-time*, RETRY=*retries*) ]**

Attempts to mount the filesystem at least once in background mode. If the first mount attempt fails, it informs you and keeps retrying after an optionally specified time delay and number of retries. If omitted, the DELAY defaults to 30 seconds (::30 in delta time). The maximum delay period you can specify is approximately 49 days. The default RETRY time value is 10. If you specify RETRY=0, the client uses 1 instead.

**/CACHE\_TIMEOUT [= ( [DIRECTORY=*t*] [ ,ATTRIBUTE=*t*] [ ,READ\_DIRECTORY] ) ]**

Caching timeout information for the mount point. The following keywords apply:

The DIRECTORY timer	Specifies the amount of time ( <i>t</i> ) the client waits between rereading a directory's status or contents. Specify the time in delta format (see <i>Delta Time Examples</i> ). The default is 30 seconds (: : 30 in delta time).
The ATTRIBUTE timer	Specifies the amount of delta time ( <i>t</i> ) the client waits between rereading a file's attributes from the server. The default is 15 seconds (::15 in delta time)
The READ_DIRECTORY keyword	Forces the client to read the contents of the directory requested when the cache timeout occurs, rather than relying on the directory's modified time. By reading the directory contents, the client can be aware of any changes to

	the number of files within the directory even if the directory's modify time was not updated.
--	---

**/CONVERT={ STREAM\_LF | STREAM\_CRLF }**  
**/NOCONVERT**

Controls whether the Client should convert sequential, variable-length, carriage return carriage control (VAR-CR) files to STREAM-LF files for UNIX system servers or STREAM\_CRLF for PC system servers. Some OpenVMS applications require that certain files remain VAR-CR. The default is /CONVERT=STREAM\_LF unless you use /SERVER\_TYPE=VMS\_SERVER, in which case MultiNet forces a /NOCONVERT.

You can only convert files opened using RMS sequential access to STREAM-LF or STREAM\_CRLF format when written by the client.

The NFS client does not perform conversions when “block mode transfers” are performed. COPY and EDT use block mode transfers when copying or creating files. Instead of COPY, use the CONVERT command. Instead of EDT, use the TPU command. Most applications do RMS sequential access when they create files on the export and these will be converted.

**/DATA=[ ( ] read-bytes [ , write-bytes ] [ ) ]**

Largest amount of NFS data received (read-bytes) or transmitted (write-bytes) in a single network operation. The default for both is 8192 bytes, the maximum allowable value appropriate for most servers. The minimum is 512. If you specify only one value, that value applies to both read and write. However, you can use different values for each.

You do not normally need to use the /DATA qualifier unless a remote server imposes a restriction on data size. Also, if the NFS server requests a smaller transfer size than the one set with this qualifier, the server's requested value will override the one set by /DATA.

**/FILEIDS={UNIQUE | NONUNIQUE}**

With UNIQUE (the default), the client uses filenames and 32-bit NFS file IDs when processing the directory information returned by the server, to determine whether cached information is valid.

With NONUNIQUE, the client uses file handles instead of file IDs in retrieving directory information. This can refresh directory entries in the client's cache more quickly, resulting in fewer "no such file" errors. However, this can degrade performance since the client must issue additional RPC requests.

`/FILEIDS=NONUNIQUE` automatically implies a `/LOOKUPS`, so do not use it together with an explicit `/NOLOOKUPS`.

### **`/FORCE`**

Force an overmount or a mount that can cause filesystem occlusion. This qualifier requires `OPER` privilege. Overmounting a `/SYSTEM` mount requires `SYSNAM` privilege. Overmounting a `/GROUP` mount requires `GRPNAM` privilege.

### **`/GID=gid`**

Default GID if no GID mapping exists for file access. The default value is `-2`. Requires `OPER` privileges.

### **`/GROUP`**

Places the logical name in the group logical name table. If the mount is the first group or system mount on the volume, `/GROUP` marks the volume as group-mounted and increments the mount count. Requires `GRPNAM` privilege. Do not use with `/SYSTEM`.

### **`/LABEL=volume-label`**

ODS-2 volume label used for the remote pathname. You can use this qualifier to provide a unique volume label on a system where there is a conflict. The default is the first 12 characters of the combined `server:mountpoint` parameter. The client accepts only the first 12 characters for all other entries. The client applies the `/LABEL` qualifier on the first mount of an NFS device only and ignores it with subsequent mounts on that device.

### **`/LOCK`**

Specifies that the client should use advisory network file locking by way of the Network Lock Manager (NLM) to coordinate access to server files.

### **`/NOLOOKUPS`**

### **`/LOOKUPS`**

With `/NOLOOKUPS` (the default), the client does not look up file handles when building directory caches. However, when accessing an individual file, it does look up its file handle; and with a directory

operation, it still looks up the handle for every file in the directory. Do not use an explicit `/NOLOOKUPS` together with `/FILEIDS=NONUNIQUE`.

### **`/NFS={ 2 | 3 }`**

Specifies that only a particular version of NFS be used when attempting to mount the unit. If this qualifier is not specified, then NFSv3 is attempted first and then NFSv2 if that fails. The NFSv3 ACP can only be used to service NFSv3 mount points and the NFSv2 ACP can only be used to service NFSv2 mount points, so caution is advised when using the `/PROCESSOR` qualifier. NFSv3 mount points will be presented as an ODS-5 disk for OpenVMS systems that recognize ODS-5 when the server maintains the case of filenames and maintains the number of hard links. When the device is presented as an ODS-5 device there is no mapping of filenames; case sensitivity and parsing rules are controlled by the VMS process parameters.

### **`/NOREADDIRPLUS`**

For NFSv3 this disables the use of the `READDIRPLUS` command to read directory and file information. The client will fall back to using `READDIR` if it detects that the server does not support `READDIRPLUS`, so this is only necessary if there is a problem when using `READDIRPLUS`. Note that `READDIRPLUS` is generally more efficient than `READDIR`.

### **`/OWNER_UIC=uic`**

Specifies the UIC assigned ownership of the volume while you mount it, thereby overriding the ownership recorded on the volume. The client applies the `/OWNER_UIC` qualifier on the first mount of an NFS device only and ignores it with subsequent mounts on that device.

### **`/PROCESSOR={UNIQUE | SAME:nfs-device | FILE:filespec}`**

Requests that `NFSMOUNT` associate an Ancillary Control Process (ACP) to process the volume, which overrides the default manner in which the client associates ACPs with NFS devices. The qualifier requires `OPER` privilege. The possible keyword values are:

UNIQUE	Creates a new ACP (additional address space) for the new NFS device. This is useful for mounting large remote filesystems so that you can accommodate more cached information. (See <i>Cache Space</i> .)
--------	---



SAME: <i>nfs-device</i>	Uses the same ACP as the specified device. The <i>nfs-device</i> specified cannot be mounted as UNIQUE. Care should be taken when using this as NFSv2 and NFSv3 mount points cannot share an ACP.
FILE: <i>filespec</i>	Creates a new ACP running the image specified by a particular file. You cannot use wildcards, node names, and directory names in the <i>filespec</i> . Requires CMKRNL or OPER privilege.

***/PROTECTION=protection-code***

Protection code assigned the volume, following the standard syntax rules for specifying protection. If you omit a protection category, the client denies that category of user access. The default is (S:RWED,O:RWED,G:RWED,W:RWED).

The client applies the /PROTECTION qualifier on the first mount of an NFS device only and ignores it with subsequent mounts on that device. /PROTECTION requires OPER privilege.

***/RETRIES=max-retries***

Maximum number of times the client retransmits an RPC request. The default is zero (0), where the client retries the request indefinitely.

***/SERVER\_TYPE=server-type***

Type of server from which the client mounts data. The valid values for *server-type* are:

UNIX  
VMS\_SERVER  
IBM\_VM

The default is either UNIX or VMS\_SERVER (if the server runs MultiNet's server).

With /SERVER\_TYPE=VMS\_SERVER, MultiNet forces /NOCONVERT and /ADF=CREATE regardless of their specified settings.

***/SHARE***

Places the logical name in the job logical name table and increments the volume mount count regardless of the number of job mounts. When the job logs out, all job mounts are dismounted, causing the volume mount count to be decremented. (See *Shared*.)

***/SUPERUSER=uid***

Has the client map users with `SYSPRV`, `BYPASS`, or `READALL` privileges to the superuser UID. The server must allow superuser access. The normal superuser UID is 0.

***/SYSTEM***

Places the logical name in the system logical name table (the default action). If the mount is the first group or system mount on the volume, this marks the volume as system mounted and increments the volume mount count. Requires `SYSNAM` privilege. Do not use with `/GROUP`.

***/TIMEOUT=timeout-period***

Minimum timeout period (in OpenVMS delta time) for initial RPC request retransmissions. The default is `::1` (one second).

The *timeout-period* value should reflect the estimated typical round-trip time for RPC requests. For slower speed links (like NFS traffic over SLIP or WANs), a larger value than the default would be appropriate.

For example, for a maximum read/write size of 8192 (see the `/DATA` qualifier) over a 19,200-baud SLIP line, the absolute minimum timeout value should be:

$$\frac{10240 \text{ bytes} * 8 \text{ bits per byte}}{19200 \text{ bits per second}} = 4.27 \text{ seconds}$$

The 10240 bytes are 8192 data bytes plus the worst case RPC overhead of 1048 bytes. Since 4.27 seconds is the absolute minimum, a more realistic value for this link would be in the range of 15 to 30 seconds to allow for other traffic.

***/TRANSPORT=protocol-type***

Network protocol used to transfer the data. The valid values are `TCP` and `UDP` (the default).

***/UID=uid***

Default UID, if no UID mapping exists for file access. The default value is -2. Requires `OPER` privileges.

***/USER=username***

Existing OpenVMS account to which the client maps unknown UIDs. The default is the USER account. If the client does not find the USER account, the DECNET account becomes the default. If the client does not find the DECNET account, [200, 200] becomes the default.

#### **/VERSION**

#### **/NOVERSION**

Use the /NOVERSION qualifier to enforce a limit of one version on a file. This is a way of imposing an NFS file versioning scheme on OpenVMS files. /VERSION, allowing multiple versions, is the default. This qualifier is disabled if connected to a MultiNet NFSv3 server. (See *Limiting File Versions*.)

#### **/WRITE**

#### **/NOWRITE**

Allows that you mount the filesystem either with write access (/WRITE) or read-only (/NOWRITE) on the local machine. If /NOWRITE, file creation, deletion, and other modifications are not allowed.

/WRITE is set by default.

## **Examples**

1. In this example, the client mounts the /usr filesystem from sigma onto the OpenVMS mount point when it references the pathname. The client keeps the path mounted until the client reaches an inactive period of 10 minutes, after which it unmounts the pathname. Subsequent references cause the client to remount the filesystem.

```
$ NFSMOUNT SIGMA "/usr" NFS0: /AUTOMOUNT=(INACTIVITY=00:10:00)
```

2. This example shows an overmount. The second mount specifies a lower level in the server path.

```
$ NFSMOUNT SIGMA "/usr" NFS1:[USERS.MNT]
%NFSMOUNT-S-MOUNTED, /usr mounted on NFS1:[USERS.MNT]
$ NFSMOUNT SIGMA "/usr/users" NFS1:[USERS.MNT] /FORCE
%NFSMOUNT-S-REMOUNTED, _NFS1:[USERS.MNT] remounted as /usr/users on SIGMA
```

3. This example shows an occluded mount. The mount point specification is "backed up" one subdirectory on the second mount. Both mounts are visible in an NFSMOUNT/SHOW. However, if you do a directory listing on NFS2:[USERS.SMITH], the [MNT] directory is no longer visible. To make the directory visible again, dismount NFS2:[USERS.SMITH].

```
$ NFSMOUNT SIGMA "/usr" NFS2:[USERS.SMITH.MNT]
```

```
%NFSMOUNT-S-MOUNTED, /usr mounted on NFS2:[USERS.SMITH.MNT]
$ NFSMOUNT SIGMA "/usr" NFS2:[USERS.SMITH] /FORCE
%NFSMOUNT-S-MOUNTED, /usr mounted on NFS2:[USERS.SMITH]
-MULTINET-I-OCCLUDED, previous contents of NFS2:[USERS.SMITH] occluded
```

---

# NFSMOUNT /CONFIG

Mounts one or more remote NFS directories based on information in a configuration file. In this way, you can maintain a regular list of server filesystems that you can automatically mount using one command.

## DCL Format

```
$ CNFSMOUNT /CONFIG=filespec
```

## Parameter

### *filespec*

OpenVMS file containing the configuration information. The contents of the file should include line entries in the format prescribed by the NFSMOUNT command:

```
server "nfs-path" mountpoint [logical] [qualifiers]
```

The configuration file must have complete information for a mount on each line (continuation lines are not allowed). The client ignores blank or comment lines. Mount requests in the file can have further configuration file references, although there is limited nesting of these requests.

## Qualifiers

**Note:** The client uses qualifiers specified with the NFSMOUNT /CONFIG command as defaults for mount requests in the configuration file. However, qualifiers included with mount requests in the file override these defaults.

See the NFSMOUNT command for details on the following qualifiers:

```
/ACP_PARAMS=( [BUFFER_LIMIT=limit-value]  
  [,DUMP]  
  [,IO_DIRECT=value]  
  [,IO_BUFFERED=value]  
  [,MAX_WORKSET=pages]  
  [,PAGE_FILE=filespec]
```

[ , PRIORITY=*base-priority*]  
[ , WORKSET=*pages*]

/ADF=*option*  
/NOADF

/AUTOMOUNT [= (INACTIVITY=*inactive-time*) ]

/BACKGROUND [= (DELAY=*delay-time*, RETRY=*retries*) ]

/CACHE\_TIMEOUT [= ( [ DIRECTORY=*t*] [ , ATTRIBUTE=*t*] ) ]

/CONVERT={ STREAM\_LF | STREAM\_CRLF }  
/NOCONVERT

/DATA=[ ( ) read-bytes [ , write-bytes ] ( ) ]

/FILEIDS={UNIQUE | NONUNIQUE}

/FORCE  
/NOFORCE

/GID=*gid*

/GROUP

/LABEL=*volume-label*

/LOCK  
/NOLOCK

/LOOKUPS  
/NOLOOKUPS

/OWNER\_UIC=*uic*

/NFS=*version*

/PROCESSOR=*keyword*

/PROTECTION=*protection-code*

/RETRIES=*max-retries*

/SERVER\_TYPE=*server-type*

/SHARE

```
/SUPERUSER=uid
/NOSUPERUSER

/SYSTEM

/TIMEOUT=timeout-period

/TRANSPORT=protocol-type

/UID=uid

/USER=username

/WRITE
/NOWRITE
```

## Examples

1. The following command consults the `CONFIG_NFS.TXT` file for mounting information.

```
$ NFSMOUNT /CONFIG=CONFIG_NFS.TXT
```

2. The following command also sets data size and username parameters (which can be overridden by qualifiers in the configuration file).

```
$ NFSMOUNT /CONFIG=CONFIG_NFS.TXT /DATA=512 /USER=BART
```

---

# CNFSMOUNT /SHOW

Displays the mounted directories at all mount points or at a particular mount point.

## Format

```
$ CNFSMOUNT /SHOW [mountpoint | device:]
```

## Parameters

### *mountpoint*

Full NFS device name and directory tree for which to show mount information. For example:

```
NFS1: [USER.NOTES]
```

Alternately, you can use a logical name for the mount point.

### *device:*

NFS device name part of the *mountpoint* parameter (such as NFS1:).

Alternately, you can use a logical name for the mount point. With the /ALL qualifier, the client uses only the device portion of the logical name.

## Qualifiers

### /ALL

Shows mount information for all servers, or a specified server or NFS device.

### /FULL

Displays the full, current operating parameters related to each mount.

See the NFSMOUNT command for descriptions of the qualifiers that correspond to each of the operating parameters.

### /QUOTA

Displays quota information for the current user's mount. The qualifier used by itself shows four columns at the top of the display indicating the block usage, soft limit (quota), hard limit, and grace period.



Use /QUOTA with the /FULL qualifier to show four additional columns indicating any possible file quotas. These show as zeros for an OpenVMS system but as actual values for UNIX systems that support file quotas.

Use /QUOTA with the /USER qualifier to request quotas for other than the default user.

### **/USER=username**

Use with /QUOTA to show quotas for a specific user. This requires the mount to have been performed using the /SUPERVISOR qualifier, which maps users with SYSPRV, BYPASS, or READALL privileges to the superuser UID. /USER requires SYSPRV or GRPPRV privileges.

## **Examples**

1. This example provides the default command display.

```
$ NFSMOUNT /SHOW
_NFS1:[000000] automount (inactivity timer 0 00:23:00.00), mounted
SIGMA.EXAMPLE.COM:/usr
_NFS2:[000000] mounted
IRIS.EXAMPLE.COM:/usr/users
```

2. This example shows characteristics of all mounts on a specific NFS device.

```
$ NFSMOUNT /SHOW NFS0: /ALL
_NFS1:[A.B] mounted
SIGMA.EXAMPLE.COM:/usr
_NFS2:[A.C] mounted
SIGMA.EXAMPLE.COM:/work
```

3. This example shows the full mount display with all operating parameters for a specific NFS device. Note that you can either enable or disable Writing and Write conversion.

```
$ NFSMOUNT /SHOW NFS1: /FULL
_NFS1:[000000] mounted
MERAK.EXAMPLE.COM:/eng/nfsuser
Transport                UDP      Writing           Enabled
Read/write size          8192/8192 Write conversion  Disabled
RPC timeout              0 00:00:01.00 ADF usage        USE,UPDATE,CREATE
RPC retry limit          0       Fileids          Unique, Nookups
Attribute time           0 00:00:15.00 Server type      TCPware, NFSv2
Directory time           0 00:00:30.00 Advisory Locking Disabled
```

```
Cache Validation    MODIFY TIME    Default user      [USER]
Superuser           No             Default UID,GID   100,15
```

4. This example shows the additional full block and file quotas for the user's mount.

```
$ NFSMOUNT /SHOW NFS2: /QUOTA /FULL
_NFS2:[000000] mounted
viola:/pctest
Disk Quotas for user [SMITH]: (inactive)
Blocks  Quota  Limit  Grace  Files  Quota  Limit  Grace
117355  500000  600000          0      0      0
Transport                                UDP Writing Enabled
Read/write size          8192/8192 Write conversion Disabled
RPC timeout              0 00:00:01.00 ADF usage USE,UPDATE,CREATE
RPC retry limit          0 Fileids Unique, Nolookups
Attribute time           0 00:00:15.00 Server type MultiNet, NFSv2
Directory time           0 00:00:30.00 Advisory Locking Disabled
Cache Validation    MODIFY TIME    Default user      [USER]
Superuser           No             Default UID,GID   100,15
```

---

# 26. Using the NFS Client

This chapter describes how to configure and maintain the MultiNet NFS client, which allows users of OpenVMS client computers to access files on a variety of server computers.

This chapter refers to the MultiNet NFS client and NFS server software as the *NFS client* and *NFS server* and the OpenVMS client system and server system as the *client* and *server*.

## Understanding the NFS Client

The MultiNet NFS client is an OpenVMS implementation of the Network File System (NFS) protocol. It allows client hosts running the OpenVMS operating system to remotely access files on a variety of server computers that use different operating systems. To users on an OpenVMS client system, all mounting and access operations are transparent, and mounted directories and files appear as native FILES-11 volumes.

## Servers and Clients

The MultiNet NFS client software allows an OpenVMS system to access file systems made available to the network by many types of server systems, including:

- UNIX/Linux
- Microsoft Windows
- OpenVMS systems running the MultiNet NFS server

The client identifies each file system by the name of its mount point on the server, which is the name of the device or directory at the top of the file system hierarchy. When mounting the file system, the client connects the mount point to a mount device in its own hierarchy; for example, NFS2 : . Through this connection, all files below the mount point are available to client users as if they resided on the mount device.

The client converts all mounted directory and file structures, contents, and names to the format required by OpenVMS automatically. For example, a UNIX file named:

```
/usr/joe/.login
```

appears to an OpenVMS client user as:

```
DISK$UNIX: [USR.JOE] .LOGIN; 1
```

The MultiNet NFS client can convert most valid UNIX file names to valid OpenVMS names and vice versa. See the *Storing OpenVMS File Names on an NFS Server* section for a complete description of the MultiNet NFS client file-naming conventions.

## MultiNet NFS Client Use of User IDs

NFS server systems identify each of their users to the network by a pair of UNIX or UNIX-style user ID (UID) and group ID (GID) codes. During an access operation, the client translates back and forth between the user's OpenVMS UIC (user identification code) and UID/GID pair.

For example, a user named Moore has an account on a UNIX server with a UID of 504 and a GID of 10. The UID and GID are mapped on the OpenVMS client to a user name BMOORE with a UIC consisting of the userid 504 and group affiliation 10.

When the NFS client ACP (ancillary control process) starts, it reads the `NFS.CONFIGURATION` file, including the UID translations list. The client uses the list to translate each OpenVMS user name to its UID/GID pair and builds a translation table that maps UID/GID pairs to their corresponding OpenVMS UICs.

As described in the following sections, you must create and maintain the UID translation list that maps each user's OpenVMS user name to a UID/GID pair. For file protections to work properly, mappings must be both unique and consistent in each direction (see the *Grouping NFS Client Systems for UID/GID Mappings* section for a description of exceptions to this rule). You cannot map a single UID to multiple OpenVMS user names, nor can you use a single user name for multiple UIDs.

Whenever the UID/GID to OpenVMS UIC mapping is modified, the NFS client must be reloaded for the changes to take effect. See the *Reloading the NFS Client* section for more information on restarting the NFS client.

## Grouping NFS Client Systems for UID/GID Mappings

If all the systems in your environment share the same UID/GID pairs, you need not create or specify NFS groups. All translations are automatically placed in the default group, which has no group name associated with it.

In the database that translates between UID/GID pairs and OpenVMS user names, each entry is associated with a particular NFS group. An NFS group is a collection of NFS systems sharing a single set of UID/GID pairs. An example of a collection of systems that would be placed in an NFS group would be a UNIX file server and diskless UNIX client systems which share the same `/etc/passwd` file. Within an NFS group, the mapping between UID/GID pairs and OpenVMS user names must be

one-to-one: you *cannot* map a single UID/GID to multiple user names, nor can you use a single user name for multiple UID/GIDs. However, duplicate translations may exist between NFS groups.

When the NFS client sends an NFS request to a server, it consults the local NFS group database to determine with which group the server is associated. If the server is not specified explicitly in a group, it is assumed to be in the default group. Once the NFS client has determined the NFS group to which the server belongs, it uses the UID/GID translation list for that group to determine the UID/GID pair to use for a particular local user when accessing files on the server.

## Mapping Example

Consider the following example. At Example, Inc., the engineering department has a group of UNIX hosts, the sales department has a collection of PCs, and the marketing department has a mix of PCs and UNIX hosts. Each group also has its own UNIX system acting as an NFS server for the group. Unfortunately, the groups did not coordinate with each other when they assigned user names and UID/GID pairs; and none of the groups are willing to change their current configurations. The accounting department, on the other hand, recently purchased a VAX-4000 computer running OpenVMS and the NFS client and wishes to use NFS to access certain personnel data that each department maintains on a local server.

The accounting system manager configures the NFS client on the OpenVMS system as follows:

1. Using the `NFS-CONFIG ADD NFS-GROUP` command, the system manager creates the three NFS groups `ENGINEERING`, `SALES`, and `MARKETING`, placing the NFS systems in each department in the appropriate NFS group (the default group will be used for systems in the accounting department).
2. Departments create accounts (and hence UID/GID pairs) on their servers that the system manager can map to local OpenVMS user ids.
3. Finally, the system manager uses the NFS Configuration Utility `ADD UID-TRANSLATION` and `ADD NFS-PASSWD-FILE` commands to create mappings between OpenVMS user names and UID/GID pairs for each NFS group. See the *Adding and Deleting Mappings* section for details on specifying these mappings.

## Effects of Incomplete Mappings

When mappings are incomplete or nonexistent, access operations are denied or severely limited.

If any server files or directories are owned by a UID for which there is no mapping, the client handles them as though they were owned by the OpenVMS user `DEFAULT ([200,200])`. The client grants access only according to the `WORLD` file protection setting of these files.

## File System Limitations

If your MultiNet system is running both the NFS client and NFS server, you cannot configure the server to export a file system that has been mounted by the client. The server can export *local disks only* when the server and client are running on the same system.

Because the NFS client does not completely emulate the "on disk" structure of the OpenVMS file system, some applications that directly read the file system may not work correctly over the NFS client.

**Note:** The NFS client can only mount file systems from OpenVMS systems that are using the NFS server.

Finally, you may notice that a file's ID (FID) changes every time the file system is remounted. This happens because the NFS protocol does not allow a local file ID to be stored on the remote host; therefore, each NFS client device (NFSx:) has its own idea of what the FID is. The client keeps a cache of local FIDs which it generates by choosing monotonically increasing numbers. Therefore, a file mounted multiple times in a cluster may have a different FID on each node. This might cause trouble with print queues that execute on a node other than the one that submitted the job.

## DISKQUOTA Limitations

Although the NFS client supports NFS server disk quotas, it does not support use of the `DCL SHOW QUOTA` command or the `DISKQUOTA` utility to examine or manipulate these quotas.

## Security and File Protections

The NFS client supports the standard OWNER, GROUP, and WORLD protections allowing READ, WRITE, EXECUTE, and DELETE access (DELETE access is taken from the WRITE access settings on the server system). Each user has a user account on the OpenVMS client as well as on the server. The NFS client compares the UIC of the user to the owner and protection mask of the directory or file and then grants or denies access, as indicated earlier in the *MultiNet NFS Client Use of User IDs* section.

OpenVMS Access Control Lists (ACLs) are supported when accessing files on NFS server OpenVMS systems, as well as most UNIX NFS server systems.

# Storing OpenVMS File Attributes on an NFS Server

The NFS protocol assumes an underlying file system in which files are merely streams of bytes with records delimited by linefeed characters (corresponding to the OpenVMS RMS Stream\_LF record format). The NFS client supports storage of non-Stream\_LF files on an NFS server through attribute description files and file name-dependent attribute defaults.

When using the NFS client, if you create a non-Stream\_LF OpenVMS file or a file with ACLs associated with it on an NFS server, the NFS client automatically creates a companion file to hold the attributes. The companion file is a text file in FDL (File Description Language) format.

The client hides the companion file from the user's view; the user sees only a single file with all of the attributes. If you rename or delete the original file from the client, the client automatically renames or deletes the companion file. However, if you rename or delete a file from the server side, you must also rename the companion file. If you do not, file attributes will be lost, the file will revert to stream attributes, and its contents may become unusable.

For example, if you create the remote indexed sequential file `foo.bar`, the client creates a second remote file `.$fdl$foo.bar` to hold the attributes. For details on controlling the NFS client's use of FDL files, see the *Advanced NFS Client Mount Options* section.

Ordinary text files (Stream\_LF files) are stored in UNIX byte-stream format and do not require companion files. If you use the OpenVMS `COPY` command to copy a non-Stream\_LF file to an NFS client mounted disk, the file will be converted automatically to Stream\_LF format. To disable this conversion, use the `NOSTREAM_CONVERSION` option of the `NFSMOUNT` `/SEMANTICS=qualifier`.

**Note:** When communicating with a UNIX NFS server system, this option prevents UNIX users from accessing these unconverted files as Stream\_LF text files.

Certain types of files default to a type other than Stream\_LF, and the absence of a companion file implies attributes other than Stream\_LF. For example, a `*.EXE` file defaults to a fixed length 512-byte record file. The below table lists the default file attributes included with MultiNet.

File Name	File Type	Default File Attributes
*.EXE	Executable	Fixed 512-byte records

* .OBJ	Object File	Variable-length records
* .OLB	Object Library	Fixed 512-byte records
* .MAI	Mail Folder	Indexed file, variable-length records
* .MLB	Macro Library	Fixed 512-byte records
* .HLB	Help Library	Fixed 512-byte records
* .TLB	Text Library	Fixed 512-byte records
* .STB	Symbol Table	Variable-length records
* .DECW\$BOOK	Bookreader Book	Variable-length records
* .DECW\$FONT	Bookreader Font	Sequential file, FORMAT undefined
* .DECW_BOOK	ULTRIX Book	Variable-length records
* .UID	DECwindows UID	Fixed 4096-byte records w/ CR

## Storing OpenVMS File Names on an NFS Server

The NFS client uses a special file-naming convention to provide a one-to-one mapping between UNIX and OpenVMS file names. As a result of this convention, there are certain restrictions on names that can be assigned to files accessed using the NFS client.

The NFS client attempts to give OpenVMS users access to all files on servers, even when server file names contain characters not permitted by OpenVMS. To accomplish this, the NFS client performs a mapping between OpenVMS and NFS server file names, using the inverse mapping of the NFS server. This mapping ensures consistency between other NFS clients accessing and creating files using the NFS server, and the NFS client accessing and creating files using other NFS servers. All mapping sequences on the OpenVMS client begin with the escape character "\$".

As "\$" is the mapping sequence escape character, a real "\$" in a file name on the server is mapped to "\$\$" on the OpenVMS client. For example, the server file name `foo$bar.c` would map to `FOO$$BAR.C` on the OpenVMS client.



A "\$" followed by a letter (A to Z) in a file name on the client indicates a case-shift in the file name on the server. For server systems like UNIX, which support case-sensitive file names, a file name can begin in lowercase and alternate between uppercase and lowercase. For example, the server file name aCaseSENSITIVEFilename would map to A\$C\$ASE\$SENSITIVEF\$ILENAME on the OpenVMS client. A "\$" followed by any digit 4 to 9 indicates a mapping as shown below.

VMS Char.	Server Char.	Hex Value	VMS Char.	Server Char.	Hex Value	VMS Char.	Server Char.	Hex Value
\$4A	^A	1	\$5A	!	21	\$7A	Space	20
\$4B	^B	2	\$5B	"	22	\$7B	;	3B
\$4C	^C	3	\$5C	#	23	\$7C	<	3C
\$4D	^D	4	\$5E	%	25	\$7D	=	3D
\$4E	^E	5	\$5F	&	26	\$7E	>	3E
\$4F	^F	6	\$5G	`	27	\$7F	?	3F
\$4G	^G	7	\$5H	(	28			
\$4H	^H	8	\$5I	)	29	\$8A	@	40
\$4I	^I	9	\$5J	*	2A	\$8B	[	5B
\$4J	^J	A	\$5K	+	2B	\$8C	\	5C
\$4K	^K	B	\$5L	'	2C	\$8D	]	5D
\$4L	^L	C	\$5N	.	2E	\$8E	^	5E
\$4M	^M	D	\$5O	/	2F			
\$4N	^N	E	\$5Z	:	3A	\$9A	`	60
\$4O	^O	F				\$9B	{	7B
\$4P	^P	10	\$6A	^@	00	\$9C		7C
\$4Q	^Q	11	\$6B	^[	1B	\$9D	}	7D
\$4R	^R	12	\$6C	^\	1C	\$9E	~	7E
\$4S	^S	13	\$6D	^]	1D	\$9F	DEL	7F
\$4T	^T	14	\$6E	^^	1E			
\$4U	^U	16	\$6F	^_	1F			
\$4V	^V	16						

\$4W	^W	17						
\$4X	^X	18						
\$4Y	^Y	19						
\$4Z	^Z	1A						

The digit after the dollar sign and the trailing letter indicates the character in the server file name. In the special case of the "dot" character (.), the first dot in the server file name maps directly to a dot in the client OpenVMS file name. Any following dot characters on the server are mapped to the character sequence \$5N on the OpenVMS client. In directory files, any dot character in the server file name maps to \$5N on the client. For example, the server file name `foo.bar#1.old` maps to `FOO.BAR$5C1$5NOLD` on the OpenVMS client (unless `foo.bar#1.old` is a directory file, in which case it maps to `FOO$5NBAR$5C1$5NOLD.DIR` on the OpenVMS client).

The NFS client also supports OpenVMS file version numbers. If a file created using the NFS client has a file version number other than 1, the resulting file on the server contains the OpenVMS version number. The highest version of the file is hard-linked to the name without the version number.

Finally, a "\$" followed by a three-digit octal number indicates a character in the file name on the server that has the binary value of that three-digit octal number. As all character binary values from 0 to 177 (octal) already have mappings, only characters from 200 to 377 are mapped in this fashion. Thus, the leading digit of the octal number must be either 2 or 3.

## NFS Client Architecture

The NFS client consists of a device driver and an ACP process that receives requests from the OpenVMS Record Management Services (RMS) and translates them into NFS requests. Because the NFS client is called by RMS, applications using RMS or the standard input/output routines of OpenVMS programming languages do not need to be modified to access files through the NFS client. The NFS client presents a \$QIO interface identical to the interface documented in the *VMS I/O User's Reference Manual: Part I*.

The NFS client includes two top-level protocols that run parallel to each other above a stack of lower-level protocols:

- The Network File System (NFS) protocol is an IP-family protocol that provides remote file system access, handling client queries.

- The Remote Procedure Call (RPC) mount protocol, RPCMOUNT, is used by the NFSMOUNT and NFSDISMOUNT commands to get mount-point information from the server systems.

Underlying the NFS and RPCMOUNT protocols is a stack of protocols:

- The remote Procedure Call (RPC) protocol allows the client to make procedure calls across the network to servers.
- The external Data Representation (XDR) protocol handles architectural differences between the client and server systems, allowing the NFS protocol to communicate between systems with dissimilar architectures.
- The RPC/NFS Lock Manager protocol allows the NFS client to support file-locking (exclusive write access).
- User Datagram Protocol (UDP), Transmission Control Protocol (TCP), and Internet Protocol (IP) are used for the lowest levels of communication.

Traditionally, NFS has only run over UDP. The NFS client also supports communication over TCP. This may provide reliability and performance improvements when communicating with NFS server systems across slow network links or wide area networks (WANs), which suffer from packet loss and delay.

## NFS Client Setup Overview

Configuring the NFS client alters the NFS . CONFIGURATION file. This file includes the OpenVMS user name and the UNIX (server) UID and GID for each user of the NFS client.

The initial setup of the NFS client consists of the following operations:

1. Using NFS-CONFIG to provide the client with a basis for translating between each user's OpenVMS client and server user IDs.
2. Mounting remote file systems by using the DCL NFSMOUNT command (requires CMKRNL, SETPRV, SYSPRV, SYSNAM, ALTPRI, DETACH, ACNT, and SYSLCK privileges).
3. Reloading the client.

The following sections describe these client setup operations.

## Creating UID/GID Mappings

There are four steps to configuring the mapping between the user IDs used for each user by the OpenVMS client and the servers:

1. Invoking the NFS-CONFIG utility.
2. Displaying the current mapping data (optional).
3. Using the utility to set up or modify the mapping.
4. Exiting the utility.

## Invoking the NFS-CONFIG Utility

To invoke NFS-CONFIG, enter the following command. In response, NFS-CONFIG reads the current NFS.CONFIGURATION file.

```
$ MULTINET CONFIGURE/NFS
MultiNet NFS Configuration Utility 5.6
[Reading in configuration from MULTINET:NFS.CONFIGURATION]
NFS-CONFIG>
```

## Displaying the Current Mapping List

If you are modifying an existing client configuration, you might want to display the current UID translations list by displaying the NFS.CONFIGURATION file: The NFS-CONFIG SHOW command displays the file.

```
NFS-CONFIG>SHOW
```

The following example shows the type of information in the file. The sample file contains:

- The UID translations list
- Other information that relates only to NFS server operation

```
NFS-CONFIG>SHOW
UID Translations: VMS Username      Unix UID      Unix GID
                  -----
                   JOHN             10            15
                   BANZAI           2             40
NFS Passwd Files: MULTINET_ROOT:[MULTINET]NFS.PASSWD
NFS-CONFIG>
```

For more information about the NFS-CONFIG SHOW command, see the NFS-CONFIG chapter in the *MultiNet Administrator's Reference*.

# Adding and Deleting Mappings

There are two methods for adding and deleting mappings of user names to UID/GID pairs. You can combine these methods as needed:

- Add and delete individual mappings with `NFS-CONFIG`.
- If the system includes UNIX servers whose user names are the same as the OpenVMS user names, you can use one or more `/etc/passwd` files as the basis for multiple mappings, and add and delete those mappings in groups with `NFS-CONFIG`.

After creating or modifying the UID Translations list, restart the client to put the changes into effect, as described in the *Reloading the NFS Client* section.

## Adding and Deleting Individual Mappings

The `ADD UID-TRANSLATION` command creates an individual mapping between an OpenVMS user name and a UID/GID pair. For example:

```
NFS-CONFIG>ADD UID-TRANSLATION JOHN 10 15
```

To create a mapping between an OpenVMS user name and a UID/GID pair associated with the NFS group `MARKETING`, use a command like the following example:

```
NFS-CONFIG>ADD UID-TRANSLATION JOHN 10 15 MARKETING
```

If you are creating UID/GID pairs, each code must be a positive integer or zero, and each user must have a unique UID. A user of multiple servers must have the same UID for each of the servers or use NFS groups to group together systems sharing the same UID mappings.

To delete these mappings, use the `DELETE UID-TRANSLATION` command, for example:

```
NFS-CONFIG>DELETE UID-TRANSLATION JOHN
```

To delete a mapping associated with an NFS group, for example, the user `JOHN` in the `MARKETING` group:

```
NFS-CONFIG>DELETE UID-TRANSLATION MARKETING/JOHN
```

Remember that after creating or modifying the UID translation list, you must restart the client to make the changes take effect (see the *Reloading the NFS Client* section.).

## Adding and Deleting NFS Groups

The `ADD NFS-GROUP` command creates an NFS group. For example:

```
NFS-CONFIG>ADD NFS-GROUP SALES WHORFIN.EXAMPLE.COM, CC.EXAMPLE.COM
```

To delete a system from an NFS group, use the `DELETE NFS-GROUP` command; for example:

```
NFS-CONFIG>DELETE NFS-GROUP SALES WHORFIN.EXAMPLE.COM
```

To delete the NFS group itself and all UID/GID mappings associated with the group, use an asterisk (\*) for the host specification; for example:

```
NFS-CONFIG>DELETE NFS-GROUP SALES *
```

## Adding and Deleting Multiple Mappings

If the UNIX and OpenVMS systems use the same user names, you can use the `/etc/passwd` files from UNIX NFS servers to create multiple mappings. To create a multi-user mapping, use FTP (or another file transfer utility) to copy the applicable `/etc/passwd` files from the UNIX systems to the OpenVMS system running NFS client. Then run the NFS Configuration Utility, and use the `ADD NFS-PASSWD-FILE` command to create the mapping. For example:

```
NFS-CONFIG>ADD NFS-PASSWD-FILE MULTINET:NFS.PASSWD
```

To create a multi-user mapping associated with the NFS group `MARKETING`, you might use the command:

```
NFS-CONFIG>ADD NFS-PASSWD-FILE MULTINET:NFS.PASSWD1 MARKETING
```

**CAUTION!** If you add or delete users, or change the mapping between user name and UID/GID in the `/etc/passwd` file on an NFS server, be sure to make the same change in the NFS `passwd` file on the client.

The following example UID translations list includes both individual mappings and `passwd` file entries (excerpted from the output of a `SHOW` command).

```
NFS Passwd Files: MULTINET:NFS.PASSWD, MULTINET:NFS.PASSWD2
UID Translations: VMS Username      Unix UID      Unix GID
                   -----
                   JOHN              10            15
                   BANZAI            2             40
```

The following example UID translation list includes NFS group entries, individual mappings, and `passwd` file entries created with the NFS Configuration Utility `ADD NFS-GROUP`, `ADD UID-TRANSLATION`, and `ADD NFS-PASSWD-FILE` commands (excerpted from the output of a `SHOW` command).

```
NFS Group Name      Members
-----
ENGINEERING         control.example.com, fang.example.com
```

```
SALES                small-berries.example.com,whorfin.example.com
NFS Passwd Files: ENGINEERING/MULTINET:NFS.PASSWD, MULTINET:NFS.PASSWD2
UID Translations: VMS Username      Unix UID      Unix GID
                   -----
                   JOHN             10           15
                   BANZAI           2            40
                   ENGINEERING/MAX  30           10
                   SALES/TOMMY      30           10
```

To delete an NFS passwd file entry, use the `DELETE NFS-PASSWD-FILE` command. For example:

```
NFS-CONFIG>DELETE NFS-PASSWD-FILE MULTINET:NFS.PASSWD
```

# Mounting and Dismounting File Systems

The final step in performing the first NFS client configuration is to mount the remote file systems that you want client users to access as if they were local files. You can also modify an existing client configuration by mounting or dismounting file systems.

## Mounting a File System

Use the `NFSMOUNT` command to mount an NFS file system. `NFSMOUNT` requires `CMKRNL`, `SETPRV`, `SYSPRV`, `SYSNAM`, `ALTPRI`, `DETACH`, `ACNT`, and `SYSLCK` privileges. For example:

```
$ nfsmount sun: "/ufs" disk$sun
%NFSMOUNT-I-MOUNTED, SUN:./ufs NFS mounted on _NFS2:
$
```

The example command mounts the file system `/ufs` which is located on the server `sun` on the local mount device `_NFS2:.`

The double quotes are necessary in the sample command because of the special meaning of the slash (`/`) character in OpenVMS. The quotes are not necessary when mounting a file system exported by another OpenVMS system.

## Dismounting a File System

When you dismount a file system, you free the resources used by the NFS client. To dismount a file system, use the `NFSDISMOUNT` command:

```
$ nfsdismount mount_device
```

For example:

```
$ nfsdismount nfs2:
```

You can use either the logical name specified in the NFSMOUNT command or the actual NFS device name (such as NFS2:) in the mount\_device field of the NFSDISMOUNT command.

## Reloading the NFS Client

Before you can use a new or revised set of UID translations, you must first reload the UID mappings into the NFS client with the DCL command:

```
$ NFSMOUNT /RELOAD
```

You may also update the client's UID mappings using NFS-CONFIG with the following command:

```
NFS-CONFIG>RELOAD
```

For instructions on using the NFSDISMOUNT and NFSMOUNT commands, see the *Mounting and Dismounting File Systems* section.

**Note:** If no file systems are mounted, reloading does not work.

## Mounting File Systems During MultiNet Startup

When the START\_MULTINET.COM script executes during MultiNet startup, it checks the MULTINET: directory for the existence of a file named NFS\_MOUNT.COM. If this file exists, it will be executed in order to mount any remote file system desired by the system manager. The following example illustrates such a file:

```
$ SET NOON
$ Show Queue
'F$GetSYI("NODENAME")'_BATCH/Output=Sys$Manager:Nfs_Mount.Tmp/All
$ Open/Read File SYS$MANAGER:NFS_MOUNT.COM
$Loop:
$ Read/End=Done File Line
$ If "'F$Element(1," ",F$Edit(Line,"TRIM,COMPRESS"))'" .Eqs. "NFS_MOUNT"
-
    Then Goto Skip
$ Goto Loop
$Done:
```



```

$ Submit/User=System/Queue='F$GetSYI("NODENAME")'_ _BATCH -
  /NoPrint -
  MultiNet:NFS_MOUNT_BATCH -
  /Name=NFS_MOUNT/Log=Sys$Manager:NFS_Mount.Log
$Skip:
$ Close File
$ Delete Sys$Manager:NFS_Mount.Tmp;*

```

This DCL program submits another DCL command file, NFS\_MOUNT\_BATCH.COM, to the queue node \_BATCH to do the work after the system boots.

The following is an example of NFS\_MOUNT\_BATCH.COM:

```

$ Verify = 'f$verify(0)
$ Set Proc/Name="NFS Mounter"
$ Purge NFS_MOUNT.Log
$ SET NOON
$ Errors = 0
$!
$! Attempt to mount the CD player on NFS.EXAMPLE.COM
$!
$ IF .Not. F$GetDVI("DISK$CD","EXISTS") Then -
NFSMOUNT/VMS/TRANS=TCP/SOFT NFS.EXAMPLE.COM::DISK$CD: DISK$CD -
/VOLUME="CD_ROM"
$ If .Not. $Status Then Errors = Errors + 1
$!
$! Check the status and requeue job if necessary.
$!
$ If Errors .Eq. 0 Then Exit
$ Submit/User=System/Queue=SYS$BATCH -
  /NoPrint /Name=NFS_MOUNT -
  MultiNet:NFS_MOUNT_BATCH -
  /After="'F$CvTime("+01:00","ABSOLUTE")'"

```

## Creating ACPs (Ancillary Control Processes) for NFS Mounts

The NFS client has an NFS\_CLIENT\_ACP process that assists the driver by performing some operations that are easier to do in a separate process rather than at the driver level.

Because this ACP process is single-threaded, using a single ACP for all NFS devices has a significant drawback. If you have multiple NFS devices mounted to different computer systems and an operation hangs on one system, all of the NFS devices are affected.

Specifying NFSMOUNT /PROCESSOR=UNIQUE creates a separate ACP process for each NFS device. This allows NFS devices to function in parallel so one device does not have to wait for an NFS

operation on another device to complete. Multiple ACPs provide for multiple outstanding I/O operations on different devices.

The setting `/PROCESSOR=UNIQUE` creates a separate `NFS_CLIENT_n` process for each mount, `n` is the number of the NFS device (for example, `NFS_CLIENT_2`, which corresponds to the device `NFS2`).

The following example illustrates the use of `/PROCESSOR=UNIQUE`, creating four ACP processes (one for each device):

```
$ NFSMOUNT /PROCESSOR=UNIQUE SCROOGE::USERS: SCROOGE$USERS
$ NFSMOUNT /PROCESSOR=UNIQUE PIP::UTIL: PIP$UTIL
$ NFSMOUNT /PROCESSOR=UNIQUE HAVERSHAM::ADMIN: HAVERSHAM$ADMIN
$ NFSMOUNT /PROCESSOR=UNIQUE MARLEY::ENG:MARLEY$ENG
```

A setting of `/PROCESSOR=SAME=nfs_device` assigns the mount to the same ACP process as the specified `nfs_device`. For example, `/PROCESSOR=SAME=NFS3` assigns this mount to the `NFS_CLIENT_3` ACP process.

**Note:** The specified device may be either the NFS device name itself (for example, `NFS3`), or a logical name pointing at the NFS device.

Mounts specified without the `/PROCESSOR` qualifier use the default process `NFS_CLIENT_ACP`.

Process Software recommends that you use the `/PROCESSOR` qualifier to group mounts on the remote server. If the server goes down, access to other servers is not affected. You can use the `/SOFT` qualifier to permit NFS operations to time-out instead of hanging indefinitely.

The following example illustrates the use of `/PROCESSOR=SAME`. In this example, all access to the server `SCOOBY` goes through one ACP process, and all access to `PIP` goes through another process.

```
$ NFSMOUNT /PROCESSOR=UNIQUE SCROOGE::USERS: SCROOGE$USERS
$ NFSMOUNT /PROCESSOR=SAME=SCROOGE$USERS SCROOGE::DKA100: SCROOGE$DKA100
$ NFSMOUNT /PROCESSOR=UNIQUE PIP::UTIL: PIP$UTIL
$ NFSMOUNT /PROCESSOR=SAME=PIP$UTIL PIP::FOO: PIP$FOO
```

## NFS Clients Using BACKUP

The OpenVMS `BACKUP` utility can write a saveset to an NFS-mounted disk, but the NFS client does not support specifying files on an NFS-mounted disk as the `input-specifier` in a `BACKUP` command.

BACKUP works with the NFS client in a *limited* way with the following restrictions:

- BACKUP preserves the UIC it finds on a file. If an NFS UNIX file has a UID that does not map to an OpenVMS UIC, the file is backed up as if it belonged to DEFAULT. When you restore the file, it will belong to the UNIX user nobody (UID -2, GID -2).
- BACKUP does not preserve certain bits of information associated with UNIX (such as the "sticky" or set-UID bits).
- NFS identifies UNIX files using a 32-byte file handle. However, the file handle must be presented to OpenVMS as a 6-byte FID. Because the number of possible 32-byte file handles is much greater than the number of possible 6-byte FIDs, MultiNet must implement a cached mapping scheme. This approach works well with applications that only care about FID consistency as long as the file is accessed. However, some applications (such as BACKUP) expect consistent FIDs for the life of the file.

**Note:** Using BACKUP with NFS-mounted files copies the contents of the files, but does not copy the semantics of files created from foreign operating systems. Process Software recommends backing up OpenVMS files to a remote tape via RMT (Remote Magtape Protocol) using MULTINET RMTALLOC (see Chapter 17).

## Advanced NFS Client Mount Options

The NFS client uses a number of techniques to map OpenVMS semantics such as file attributes and file version numbers to the underlying UNIX file system present on an NFS server. In using these techniques, it makes the assumption that full UNIX semantics, such as file names and hard links, are available on the NFS server. Failure to provide an NFS server that satisfies these assumptions results in incorrect operation of the NFS client.

NFS servers that do not correctly support the full UNIX file system semantics can be used with the NFS client in a restricted fashion. The /SEMANTICS qualifier to the NFSMOUNT command tells the NFS client that the NFS server does not support certain operations, and these operations should not be attempted. Use of certain /SEMANTICS values will mean that some OpenVMS capabilities will not be available, but the client will still function.

The below table lists each semantic and how it changes the behavior of the NFS client. You can specify more than one semantic on a mount request. For more information on the use of the /SEMANTICS qualifier, refer to the NFSMOUNT command page in the *MultiNet Administrator's Reference*.

Semantics Value	Description
ADVISORY_CLOSE	Sends an OpenVMS server a command to close the file when there are no more references to it on the client.
NOFDL_FILES	Disables the use of <code>.\$fdl\$</code> files by the NFS client to store RMS attributes. This option must be used if the NFS server does not allow these file names. Its use severely limits the ability of the NFS client to store record attributes.
NOLINKS	Normally, the NFS client uses a hard link operation to link the top version of a file name <code>foo.bar;12</code> to the unversioned name <code>foo.bar</code> for more convenient access from the NFS server side. This option disables creating this hard link and may be used either to reduce the overhead of creating it or if the NFS server does not support hard links.
NOSTREAM_CONVERSION	Normally, the NFS client converts requests to create Variable Length Record Carriage Return Carriage Control (VAR-CR) files into requests to create Stream files, to ensure that text files can be shared between OpenVMS and UNIX systems. This option disables this conversion.
NOUNIQUE_FILENO	Specifies that the NFS server does not generate unique file numbers for each file (most NFS servers do). If the client knows that file numbers are unique, it uses a faster algorithm to refresh stale directory entries in the cache. Use of this switch disables the faster refresh algorithm.
NOVERSIONS	Normally the NFS client stores multiple versions of OpenVMS files by using the semicolon character in the file name on the NFS server side. This option must be used to disable the creation multiple versions of files if the NFS server does not support file names with the semicolon character.
NOVMS_ACCESS_CHECKING	Specifies that the client should not perform a full OpenVMS access check, including a check for ACLs and security alarms. If this option is not specified, the NFS client will consider ACLs and security alarms when granting or denying access.

PRESERVE_DATES	Specifies that the client should maintain the file's creation, backup, and expiration dates in an FDL file. For information on storing OpenVMS file attributes in FDL files, see the <i>Storing OpenVMS File Names on an NFS Server</i> section.
UPPER_CASE_DEFAULT	Assumes file names are in uppercase on the server until it sees the \$ character used to toggle case.
VMS_FILENAMES	Specifies that the NFS client does not perform the usual mapping between OpenVMS and UNIX-style file names. This option can be used to permit all OpenVMS file names to be stored using the NFS client, but its use prevents the NFS client from being used to access files which do not conform to the OpenVMS file name conventions.
VMS_SERVER	Specifies that the NFS server is an NFS server supporting protocol V3 or later, which supports the OpenVMS-specific extensions to the NFS protocol to store file attributes. If the NFS server does not support these extensions, the mount will fail. This option overrides any other semantics specified.

# 27. Configuring the Secure Shell (SSH) 1 Server

This chapter describes how to configure and maintain the MultiNet Secure Shell (SSH) v1 server.

This is the server side of the software that allows secure interactive connections to other computers in the manner of rlogin/rshell/telnet. The SSH server has been developed to discriminate between SSH v1 and SSH v2 protocols, so the two protocols can coexist simultaneously on the same system.

## SSH1 and SSH2 Differences

SSH1 and SSH2 are different, and incompatible, protocols. While SSH2 is generally regarded to be more secure than SSH1, both protocols are offered by MultiNet, and although they are incompatible, they may exist simultaneously on a MultiNet system. The MultiNet server front-end identifies what protocol a client desires to use, and will create an appropriate server for that client.

**Note:** You must install the DEC C 6.0 backport library on all OpenVMS VAX v5.5-2 and v6.0 systems prior to using SSH. This is the AACRT060.A file. You can find the ECO on the MultiNet CD the following directory: VAX55\_DECC\_RTL.DIR.

### Restrictions:

When using SSH1 to connect to a VMS server, if the VMS account is set up with a secondary password, SSH1 does not prompt the user for the secondary password. If the VMS primary password entered is valid, the user is logged in, bypassing the secondary password.

When using SSH1 to execute single commands (in the same manner as RSHELL), some keystrokes like CTRL+Y are ignored. In addition, some interactive programs such as HELP may not function as expected. This is a restriction of SSH1. If this behavior poses a problem, log into the remote system using SSH1 in interactive mode to execute the program.

# Understanding the MultiNet Secure Shell Server

Secure Shell daemon (SSHD) is the daemon program for SSH that listens for connections from clients. The server program replaces rshell and telnet programs. The server/client programs provide secure encrypted communications between two untrusted hosts over an insecure network. A new daemon is created for each incoming connection. These daemons handle key exchange, encryption, authentication, command execution, and data exchange.

## Servers and Clients

A MultiNet SSH server is an OpenVMS system server that acts as a host for executing interactive commands or for conducting an interactive session. The server software consists of two processes (for future reference, SSHD will refer to both SSHD\_MASTER and SSHD, unless otherwise specified):

- SSHD\_MASTER, recognizes the differences between SSH v1 and SSH v2 and starts the appropriate server. If the request is for SSH v1, then a new SSH v1 server is run; if the request is for SSH v2, then a new SSH v2 server is run.
- SSHD, a copy of which is spawned for each time a new connection attempt is made from a client. SSHD handles all the interaction with the SSH client.

A client is any system that accesses the server. A client program (SSH) is provided with MultiNet, but any SSH client that uses SSH version 1 protocol may be used to access the server. Examples of such programs are FISSH, MultiNet SSH, and TCPware SSH on OpenVMS systems; TTSSH, SecureCRT, F-Secure SSH Client, and PuTTY on Windows-based systems; and other SSH programs on UNIX-based systems.

## Security

Each host has a host-specific RSA key that identifies the host. Additionally, when the SSHD daemon starts, it generates a server RSA key. This key is regenerated every hour (the time may be changed in the configuration file) if it has been used, and is never stored on disk. Whenever a client connects to the SSHD daemon:

- SSHD sends its host and server public keys to the client.
- The client compares the host key against its own database to verify that it has not changed.

- The client generates a 256 bit random number. It encrypts this random number using both the host key and the server key, and sends the encrypted number to the server.
- The client and the server start to use this random number as a session key which is used to encrypt all further communications in the session.

The rest of the session is encrypted using a conventional cipher. Currently, IDEA (the default), DES, 3DES, Blowfish, and ARCFOUR are supported.

- The client selects the encryption algorithm to use from those offered by the server.
- The server and the client enter an authentication dialog.
- The client tries to authenticate itself using any of the following methods:
  - .rhosts authentication
  - .rhosts authentication combined with RSA host authentication
  - RSA challenge-response authentication
  - password-based authentication

**Note:** Rhosts authentication is normally disabled because it is fundamentally insecure, but can be enabled in the server configuration file, if desired.

System security is not improved unless the RLOGIN and RSHELL services are disabled.

When the client authenticates itself successfully, a dialog is entered for preparing the session. At this time the client may request things such as:

- forwarding X11 connections
- forwarding TCP/IP connections
- forwarding the authentication agent connection over the secure channel

Finally, the client either requests an interactive session or execution of a command. The client and the server enter session mode. In this mode, either the client or the server may send data at any time, and such data is forwarded to/from the virtual terminal or command on the server side, and the user terminal in the client side. When the user program terminates and all forwarded X11 and other connections have been closed, the server sends command exit status to the client, and both sides exit.



# Break-in and Intrusion Detection

Care must be exercised when configuring the SSH clients and server to minimize problems due to intrusion records created by OpenVMS security auditing. The SSH user should consult the system manager to determine the authentication methods offered by the SSH server. The client should then be configured to not attempt any authentication method that is not offered by the server.

If a client attempts authentication methods not offered by the server, the OpenVMS security auditing system may log several intrusion records for each attempt to create a session to that server. The result being that the user could be locked out and prevented from accessing the server system without intervention from the server's system manager.

The authentication methods to be offered by the server are determined by the configuration keywords `RhostsAuthentication`, `RhostsRSAAuthentication`, `RSAAAuthentication`, and `PasswordAuthentication`. The number of intrusion records to be logged for any attempted SSH session is determined by the `StrictIntrusionLogging` configuration keyword.

When `StrictIntrusionLogging` is set to YES (the default), each method that is tried and fails causes an intrusion record to be logged. When `Rhosts`, `RhostsRSA` or `RSA` authentications are attempted and fail, one intrusion record will be logged for each failed method.

When password authentication is attempted, one intrusion record will be logged for each failed password.

## Example 1

The server is set up to allow `Rhosts`, `RSA`, and password authentication; also, up to three password attempts are allowed. If all methods fail, five intrusion records are logged:

1 for the failed `Rhosts`

1 for the failed `RSA`

3 for the failed password attempts, one per attempt

When `StrictIntrusionLogging` is set to NO, it has the effect of relaxing the number of intrusions logged. Overall failure of all authentication methods simply counts as a single failure, except for password authentication. The following rules apply:

- When password authentication is attempted, one intrusion record is logged for each failed password.
- When any of `Rhosts`, `RhostsRSA`, or `RSA` authentication fails, and password authentication is not attempted, exactly one intrusion record is logged, as opposed to one for each failed method.

- When any of Rhosts, RhostsRSA, or RSA authentication fails, but password authentication is attempted and succeeds, the only intrusion record(s) logged is one for each failed password attempt.

### Example 2:

The server is set up to allow Rhosts, RSA, and password authentication; also, up to three password attempts are allowed. If all methods fail, three intrusion records are logged:

0 for the failed Rhosts

0 for the failed RSA

3 for the failed password attempts, one per attempt

### Example 3:

The server is set up to allow Rhosts, RSA, and password authentication; also, up to three password attempts are allowed. Rhosts and RSA fail, but password authentication is successful after 1 failed password. Therefore, one intrusion record is logged:

0 for the failed Rhosts

0 for the failed RSA

1 for the failed password attempt

### Example 4:

The server is set up to allow Rhosts, RhostsRSA, and RSA authentication, but not password authentication. If all methods fail, one intrusion record is logged.

### Example 5:

The server is set up to allow Rhosts, RhostsRSA, and RSA authentication, but not password authentication. Rhosts and RSA authentication both fail, but RhostsRSA succeeds. No intrusion records are logged.

# Configuring SSHD Master for SSH1

SSHD Master is configured using the `MULTINET CONFIGURE /SERVER` command, selecting SSH, and using the following options:

```
$ MULTINET CONFIGURE /SERVER  
MultiNet Server Configuration Utility V5.6
```

```

[Reading in configuration from MULTINET:SERVICES.MASTER_SERVER]
SERVER-CONFIG>SELECT SSH
[The Selected SERVER entry is now SSH]
SERVER-CONFIG>SET PARAM
Delete parameter "enable-ssh1" ? [NO] RETURN
Delete parameter "enable-ssh2" ? [NO] RETURN
You can now add new parameters for SSH.  An empty line terminates.
Add Parameter: PORT 33000
Add Parameter: RETURN
[Service specific parameters for SSH changed]
SERVER-CONFIG>SHOW/FULL
Service "SSH":
    INIT() = Merge_Image
    Program = "MULTINET:LOADABLE_SSH_CONTROL"
    Priority = 5
    Log for Accepts & Rejects = OPCOM
    Parameters = "enable-ssh1"
                 "enable-ssh2"
                 "port 33000"

SERVER-CONFIG>EXIT
[Writing configuration to
MULTINET_COMMON_ROOT:[MULTINET]SERVICES.MASTER_SERVER]

```

**Note:** The recommended method to start SSHD Master is to use the MULTINET NETCONTROL SSH START command. All of these options are set using MULTINET CONFIG/SERVER, and modifying the SSH service.

## Expired Passwords

The SSH v1 protocol does not provide a method for changing an expired VMS password. When an expired password is encountered by the SSH1 server, it will do one of two things.

1. If the logical name MULTINET\_SSH\_ALLOW\_EXPIRED\_PW is defined for allowing access for passwords that have exceeded the UAF value for PWDLIFETIME, or if the logical name MULTINET\_SSH\_ALLOW\_PREEXPIRED\_PW is defined for allowing access for users that have a pre-expired password, the server will allow the user to log in. In the logical name table LNM\$SSH\_LOGICALS, the logical name MULTINET\_SSH\_pid\_PWDEXP (where pid is the process ID for the user process) will be defined. The system manager can look for this logical to be defined, and if so, take action such as executing the DCL SET PASSWORD command.

2. If the appropriate logical is not set as described above, the user will be denied access to the system. In that case, the user must log in interactively via another mechanism such as telnet and change the password, or the system manager must reset the password.

When a user is allowed access to the system with an expired password, the `LOGIN_FLAGS` for the process will reflect this. The values of the `LOGIN_FLAGS` will be as follows:

- new mail has been received (`JPI$M_NEW_MAIL_AT_LOGIN`)
- the password is about to expire (`JPI$M_PASSWORD_WARNING`)
- the password has expired (`JPI$M_PASSWORD_EXPIRED`)

The DCL lexical function `F$GETJPI` may be used to examine these flags, as can the `$GETJPI (W)` system service or `LIB$GETJPI RTL` function. When an expired password value is detected, the user may then execute a `SET PASSWORD` command in the command procedure run for the account.

```
For example:
$!
$! Login_flags:
$!   1 = new mail messages waiting (JPI$M_NEW_MAIL_AT_LOGIN)
$!   4 = password expired during login (JPI$M_PASSWORD_EXPIRED)
$!   5 = password expires within 5 days (JPI$M_PASSWORD_WARNING)
$!
$ flags = f$getjpi("", "LOGIN_FLAGS")
$ new_flags = (flags/2)*2
$ if new_flags .ne. flags then write sys$output "New mail waiting"
$!
$!Note - new_flags is used below because it has the NEW_MAIL_AT_LOGIN$
$! bit stripped. The rest of the possible values are all
$! discrete; i.e., you can't have combinations of them at the
$! same time.
$!
$ if new_flags .eq. 4 then write sys$output "Password expired during login"
$ if new_flags .eq. 5 then write sys$output "Password expires within 5 days"
$!
```

## OPTIONS

### **bits** *n*

Specifies the number of bits in the server key. The default is 768.

### **ssh1-config-file** *filename*

Specifies the name of the configuration file. The default is `MULTINET:SSHD_CONFIG`.

**debug *debug-level***

Turns debugging on using any non-zero debug level.

**enable-ssh1**

Enables SSH v1 sessions.

**host-key-file *filename***

Specifies the file from which the host key is read. The default is `MULTINET:SSH_HOST_KEY`.

**keygen-time *n***

Specifies how often the server key is regenerated. The default is 3600 seconds (one hour). The motivation for regenerating the key often is that the key is never stored physically on disk. It is kept in the address space of the server, and after an hour, it becomes impossible to recover the key for decrypting intercepted communications even if the machine is broken into or physically seized. A value of zero indicates that the key will never be regenerated.

**listen-address**

Specify the IPV4 address on which to listen for connect request. This may be a valid IPV4 address or ANY to listen on all addresses. If not specified, the default is to listen on all addresses.

**port *n***

Specifies the port on which the server listens for connections. The default is 22.

**quiet\_mode**

Specifies that nothing is sent to the SSH system log. Normally, the beginning, authentication, and termination of each connection is logged.

**verbose**

Specifies that verbose message logging will be performed by SSHD MASTER.

# Configuration File

SSHD reads configuration data from `MULTINET:SSHD_CONFIG` (or the file specified with the `ssh1-config-file` keyword in `MULTINET CONFIGURE/SERVER`). The file contains keyword value pairs, one per line. The following keywords are possible. Keywords are case insensitive.

Keyword	Value	Default	Description
<code>AllowForwardingPort</code>	Port list		Permit forwarding for the specified ports
<code>AllowForwardingTo</code>	Host/port list		Permit forwarding for hosts
<code>AllowGroups</code>	List		Access control by UAF rights list entries
<code>AllowHosts</code>	Host list		Access control by hostname
<code>AllowShosts</code>	Host list		Access control by hostname
<code>AllowTcpForwarding</code>	Y/N	Y	Enable TCP port forwarding
<code>AllowUsers</code>	User list		Access control by username
<code>DenyForwardingPort</code>	Port list		Forbid forwarding for ports
<code>DenyForwardingTo</code>	Host/port list		Forbid forwarding for hosts
<code>DenyGroups</code>	Rights list		Deny access for UAF rightslist identifiers
<code>DenyHosts</code>	Host list		Deny access for hosts
<code>DenySHosts</code>	Host list		Deny access for hosts
<code>DenyUsers</code>	User list		Access control by username

FascistLogging	Y/N	Y	Verbose logging
Hostkey	Filename	Ssh_host_key.	Host key filename
IdleTimeout	Time	0 (infinite)	Set idle timeout
IgnoreRhosts	Y/N	N	Ignore local rhosts
IgnoreRootRhosts	Y/N	Y	Ignore system rhosts
KeepAlive	Y/N	Y	Send keepalives
ListenAddress	IP address	0.0.0.0	Listen on given interface
LoginGraceTime	Time	600	Time limit for authentication in seconds
PasswordAuthentication	Y/N	Y	Permit password authentication
PermitEmptyPasswords	Y/N	N	Permit empty (blank) passwords
PermitRootLogin	Y/N	N	SYSTEM can log in
QuietMode	Y/N	N	Quiet mode
RandomSeed	Filename	Random_seed	Random seed file
RhostsAuthentication	Y/N	N	Enable rhosts authentication
RhostsRSAAuthentication	Y/N	Y	Enable rhosts with RSA authentication
RSAAuthentication	Y/N	Y	Enable RSA authentication
StrictIntrusionLogging	Y/N	Y	Determine how intrusion records are created by failed authentication attempts
StrictModes	Y/N	N	Strict checking for directory and file protection

SyslogFacility	Syslog level	“DAEMON”	Syslog log facility
VerboseLogging	Y/N	Y	Verbose logging
X11Forwarding	Y/N	Y	Enable X11 forwarding
X11DisplayOffset	#offset	10	Limit X displays for SSH

# Starting the SSH Server for the First Time

Follow these instructions for using SSH for the first time.

1. Use the `MULTINET CONFIGURE/SERVER` command to enable the SSH v1 server.

```

$ MULTINET CONFIGURE/SERVER
MultiNet Server Configuration Utility V5.6
[Reading in configuration from MULTINET:SERVICES.MASTER_SERVER]
SERVER-CONFIG>SHOW/FULL SSH
Service "SSH": ***DISABLED***
    INIT() = Merge_Image
    Program = "MULTINET:LOADABLE_SSH_CONTROL"
    Priority = 5
    Parameters = "enable-ssh1"
                "enable-ssh2"

SERVER-CONFIG>ENABLE SSH
SERVER-CONFIG>EXIT
[Writing configuration to
MULTINET_COMMON_ROOT:[MULTINET]SERVICES.MASTER_SERVER]

```

**Note:** The parameter `enable-ssh1` must be set. If it is not set, SSH v1 sessions will not be accepted by the server.



2. Use SSHKEYGEN /SSH1 to generate an ssh1 key and to create the file SSH\_HOST\_KEY in the MULTINET: directory.

```
$ MULTINET SSHKEYGEN /SSH1 /HOST
Initializing random number generator...
Generating p: ...++ (distance 64)
Generating q: .....++ (distance 516)
Computing the keys...
Testing the keys...
Key generation complete.
Key file will be MULTINET_ROOT:[MULTINET]SSH_HOST_KEY.
Your identification has been saved in MULTINET:SSH_HOST_KEY.
Your public key is:
1024 37
1210318365576698697865367869291969476388228444969905611864276308
9072776904462744415966821020109463617644202397294642277946718549
4404442577594868297087171013359743853182442579923801302020844011
5343754909847513973160249324735913146330232410424936751015953611
18716872491123857940537322891584850459319961275605927
SYSTEM@roadrr.example.com
Your public key has been saved in MULTINET_ROOT:[MULTINET]SSH_HOST_KEY.pub
```

3. Copy the template configuration file to the MultiNet directory renaming it to SSHD\_CONFIG.;

```
$ COPY MULTINET:SSHD_CONFIG.TEMPLATE MULTINET:SSHD_CONFIG.;
```

**Note:** As delivered, the template file provides a reasonably secure SSH environment. However, Process Software recommends this file be examined and modified appropriately to reflect the security policies of your organization.

4. Restart MultiNet. This creates the SSH server process and defines the SSH logical names.

```
$ @MULTINET:START_SERVER_RESTART
$ SHOW_PROCESS "SSHD Master"
7-APR-2020 09:03:06.42  User: SYSTEM          Process ID:    00000057
                        Node: PANTHR         Process name:  "SSHD Master"

Terminal:
User Identifier:      [SYSTEM]
Base priority:       4
Default file spec:   Not available
Number of Kthreads: 1

Devices allocated:   BG1:
                    BG2:
```

```
$ SHOW LOGICAL/SYSTEM SSH*
```

```
(LNM$SYSTEM_TABLE)
```

```
"SSH_DIR" = "MULTINET_SPECIFIC_ROOT:[MULTINET]"
"SSH_EXE" = "MULTINET_COMMON_ROOT:[MULTINET]"
"SSH_LOG" = "MULTINET_SPECIFIC_ROOT:[MULTINET.SSH]"
"SSH_TERM_MBX" = "MBA23:"
```

## Configuring the SSH1 Server on a VMSccluster with a Common System Disk

When configuring the SSH1 server on a VMSccluster with a common system disk, you must create the appropriate directories on all cluster nodes other than one on which MultiNet was originally installed. Note that this does not need to be done for cluster members that do not share a common system disk.

The following procedure should be followed on each cluster node other than the cluster node on which MultiNet was originally installed:

- Create the necessary directory:

```
$ CREATE/DIR MULTINET_SPECIFIC[MULTINET_SSH]/PROT=(WO:RE,GR:RE)
```

- Edit the MULTINET\_SPECIFIC:[MULTINET.SSH]SSHD\_CONFIG file as necessary. This may be copied from another cluster node, or it may be created fresh from the SSHD\_CONFIG.TEMPLATE file.
- Configure the SSH1 server using MULTINET CONFIGURE/SERVER
- Generate the SSH1 host keys using MULTINET SSHKEYGEN/SSH1/HOST
- (Re)start SSHD Master using MULTINET NETCONTROL SSH RESTART

## Changing SSH1 Configuration File after Enabling SSH1

If you make a change to the SSH1 configuration file after you have enabled SSH1, you must restart SSH for these changes to take effect.

```
$ MULTINET NETCONTROL SSH RESTART
```

**Note:** When issuing the `RESTART` command for SSH, all active SSH server sessions are terminated. Active client sessions are not affected.

## Connection and Login Process

To create a session, SSHD does the following:

1. `SSHD_MASTER` process sees the connection attempt. It creates an `SSHD v1` or `v2` process, depending on the protocol version presented to it by the client. `SSHD_MASTER` then passes necessary information to the `SSHD` process, such as the server key and other operating parameters.
2. `SSHD` process performs validation for the user.
3. Assuming the login is successful, `SSHD` process creates a pseudoterminal for the user (an `_FTAnn` device). This device is owned by the user logging in.
4. `SSHD` process creates an interactive process on the pseudoterminal, using the username, priority, and privileges of the user logging in. If a command was specified, it is executed and the session is terminated.
5. SSH generates the file `SSHD.LOG` in the directory `MULTINET_ROOT: [MULTINET.SSH]` for each connection to the SSH server. Many connections result in many log files. Instead of purging the files on a regular basis, use the following DCL command to limit the number of versions:

```
$ SET FILE /VERSION LIMIT=x MULTINET_ROOT: [MULTINET.SSH] SSHD.LOG
```

**Note:** The value for `/VERSION_LIMIT` must not be smaller than the maximum number of simultaneous SSH sessions anticipated. If the value is smaller, SSH users may be prevented from establishing sessions with the server.

# FILES

## **MULTINET:HOSTS.EQUIV**

Contains host names, one per line. This file is used during `.rhosts` authentication. Users on those hosts are permitted to log in without a password, provided they have the same username on both machines. The hostname may also be followed by a username. Such users are permitted to log in as any user on the remote machine (except `SYSTEM`). Additionally, the syntax `+@group` can be used to specify netgroups. Negated entries start with a dash (-). If the client host/user is matched in this file, login is permitted provided the client and server usernames are the same. Successful RSA host authentication is required. This file should be world-readable but writeable only by `SYSTEM`.

It is never a good idea to use usernames in `hosts.equiv`. It means the named user(s) can log in as anybody, which includes accounts that own critical programs and directories. Using a username grants the user `SYSTEM` access. The only valid use for usernames is in negative entries.

**Note:** This warning also applies to `RSHELL/RLOGIN`.

## **MULTINET:SHOSTS.EQUIV**

Processed as `MULTINET:HOSTS.EQUIV`. May be useful in environments that want to run both `RSHELL/RLOGIN` and `SSH`.

## **MULTINET:SSH\_HOST\_KEY**

Contains the private part of the host key. This file does not exist when MultiNet is first installed. The `SSH` server starts only with this file. This file must be created manually using the command:

```
$ MULTINET SSHKEYGEN /SSH1 /HOST
```

This file should be owned by `SYSTEM`, readable only by `SYSTEM`, and not accessible to others.

To create a host key with a name that is different than what `SSHKEYGEN` creates, do one of the following:

- Generate with `MULTINET SSHKEYGEN /SSH1 /HOST` and simply rename the file.
- Generate a public/private key pair using `SSHKEYGEN` without the `/HOST` switch, and copying and renaming the resulting files appropriately.

By default the logical name `SSH_DIR` points to the `MULTINET_SPECIFIC_ROOT:[MULTINET]` directory.

Refer to the *MultiNet User's Guide*, Chapter 8, for more details about SSHKEYGEN.

**MULTINET:SSH\_HOST\_KEY.PUB**

Contains the public part of the host key. This file should be world-readable but writeable only by SYSTEM. Its contents should match the private part of the key. This file is not used for anything; it is only provided for the convenience of the user so its contents can be copied to known hosts files.

**MULTINET:SSH\_KNOWN\_HOSTS**

**SYS\$LOGIN:[.SSH]KNOWN\_HOSTS**

Checks the public key of the host. These files are consulted when using rhosts with RSA host authentication. The key must be listed in one of these files to be accepted. (The client uses the same files to verify that the remote host is the one you intended to connect.) These files should be writeable only by SYSTEM (the owner). MULTINET:SSH\_KNOWN\_HOSTS should be world-readable, and SYS\$LOGIN:[.SSH]KNOWN\_HOSTS can, but need not be, world-readable.

**SSH2:SSH\_RANDOM\_SEED**

**SYS\$LOGIN:[.SSH]RANDOM\_SEED**

Contains a seed for the random number generator. This file should only be accessible by SYSTEM.

**MULTINET:SSHD\_CONFIG**

Contains configuration data for SSHD. This file should be writeable by system only, but it is recommended (though not necessary) that it be world-readable.

**AUTHORIZED\_KEYS**

Located in the user's SYS\$LOGIN[.SSH] directory, this file lists the RSA keys that can be used to log into the user's account. This file must be readable by SYSTEM. It is recommended that it not be accessible by others. The format of this file is described in the next section.

**SYS\$LOGIN:.SHOSTS**

Located in the user's SYS\$LOGIN:[.SSH] directory, permits access using SSH only. For SSH, this file is the same as for .rhosts. However, this file is not used by the RLOGIN and RSHELL daemon.

## **SYS\$LOGIN: .RHOSTS**

This file contains host-username pairs, separated by a space, one per line. The given user on the corresponding host is permitted to log in without a password. The same file is used by RLOGIN and RSHELL. SSH differs from RLOGIN and RSHELL in that it requires RSA host authentication in addition to validating the hostname retrieved from domain name servers. The file must be writeable only by the user. It is recommended that it not be accessible by others. It is possible to use netgroups in the file. Either host or username may be of the form `+@groupname` to specify all hosts or all users in the group.

# **AUTHORIZED\_KEYS File Format**

The `SYS$LOGIN: [.SSH]AUTHORIZED_KEYS` file lists the RSA keys that are permitted for RSA authentication. Each line of the file contains one key (empty lines and lines starting with a # are comments and ignored). Each line consists of the following fields, separated by spaces:

<b>Key</b>	<b>Description</b>
<code>bits</code>	The length of the key in bits.
<code>comment</code>	Not used for anything (but may be convenient for the user to identify the key).
<code>exponent</code>	Used to identify and make up the key.
<code>modulus</code>	Used to identify and make up the key.
<code>options</code>	Optional; its presence is determined by whether the line starts with a number or not (the option field never starts with a number.)

**Note:** Lines in this file are usually several hundred characters long (because of the size of the RSA key modulus). You do not want to type them in; instead, copy the `IDENTITY.PUB` file and edit it. The options (if present) consists of comma-separated option specifications. No spaces are permitted, except within double quotes. Option names are case insensitive.

The following RSA key file `AUTHORIZED_KEYS` option specifications are supported:

**`allowforwardingport="port-list"`**

Can be followed by any number of port numbers, separated by spaces. Remote forwarding is allowed for those ports whose number matches one of the patterns.

You can use `*` as a wildcard entry for all ports.

You can use these formats `>x`, `<x`, and `x_y` to specify greater than, less than, or inclusive port range. By default, all port forwardings are allowed.

The quotes (`" "`) are required. For example:

```
allowforwardingport "2,52,2043"
```

**`allowforwardingto="hostname:port-list"`**

Can be followed by any number of hostname and port number patterns, separated by spaces. A port number pattern is separated from a hostname pattern by a colon. For example: `hostname:port`

Forwardings from the client are allowed to those hosts and port pairs whose name and port number match one of the patterns.

You can use `*` and `?` as wildcards in the patterns for host names. Normal name servers are used to map the client's host into a fully-qualified host name. If the name cannot be mapped, its IP address is used as the hostname.

You can use `*` as a wildcard entry for all ports.

You can use these formats `>x`, `<x`, and `x_y` to specify greater than, less than, or inclusive port range. By default, all port forwardings are allowed.

**`command="command"`**

Specifies the command to be executed whenever this key is used for authentication. The user-supplied command (if any) is ignored. You may include a quote in the command by surrounding it with a backslash (`\`). Use this option to restrict certain RSA keys to perform just a specific operation. An example might be a key that permits remote backups but nothing else. Notice that the client may specify TCP/IP and/or X11 forwardings unless they are prohibited explicitly.

**Denyforwardingport="port-list"**

Can be followed by any number of port numbers, separated by spaces. Remote forwardings are disallowed for those ports whose number matches one of the patterns.

You can use "\*" as a wildcard entry for all ports.

You can use these formats '>x', '<x', and 'x\_x' to specify greater than, less than, or inclusive port range.

**Denyforwardingto="hostname:port-list"**

Can be followed by any number of hostname and port number patterns, separated by spaces. A port number pattern is separated from a hostname by a colon. For example: *hostname:port number pattern*

Forwardings from the client are disallowed to those hosts and port pairs whose name and port number match one of the patterns.

You can use '\*' and '?' as wildcards in the patterns for host names. Normal name servers are used to map the client's host into a fully-qualified host name. If the name cannot be mapped, its IP address is used as a host name.

You can use "\*" as a wildcard entry for all ports.

You can use these formats '>x', '<x', and 'x\_x' to specify greater than, less than, or inclusive port range.

**from="pattern-list"**

In addition to RSA authentication, specifies that the fully-qualified name of the remote host must be present in the comma-separated list of patterns. You can use '\*' and '?' as wildcards.

The list may contain patterns negated by prefixing them with '!'; if the fully-qualified host name matches a negated pattern, the key is not accepted.

This option increases security. RSA authentication by itself does not trust the network or name servers (but the key). However, if somebody steals the key, the key permits login from anywhere in the world. This option makes using a stolen key more difficult because the name servers and/or routers would have to be comprised in addition to just the key.

**idle-timeout=time**



Sets the idle timeout limit to a time in seconds (s or nothing after the number), in minutes (m), in hours (h), in days (d), or in weeks (w). If the connection has been idle (all channels) for that time, the process is terminated and the connection is closed.

#### **no-agent-forwarding**

Forbids authentication agent forwarding when used for authentication.

#### **no-port-forwarding**

Forbids TCP/IP forwarding when used for authentication. Any port forward requests by the client will return an error. For example, this might be used in connection with the `command` option.

#### **no-X11-forwarding**

Forbids X11 forwarding when used for authentication. Any X11 forward requests by the client will return an error.

## **RSA Key File Examples**

```
1024 33 12121...312314325 ylo@foo.bar
from="*.emptybits.com,!sluf.pscos.com"
```

```
1024 35 23...2334 ylo@niksula
command="dir *.txt",no-port-forwarding
```

```
1024 33 23...2323 xxxxx.acme.com
allowforwardingport="localhost:80"
```

```
1024 35 23...2334 www@localhost
```

## **SSH\_KNOWN\_HOSTS File Format**

The `MULTINET:SSH_KNOWN_HOSTS` and `SYS$LOGIN:[.SSH]KNOWN_HOSTS` files contain host public keys for all known hosts. The global file should be prepared by the administrator (optional), and the per-user file is maintained automatically; whenever the user connects an unknown host its key is added to the per-user file. Each line in these files contains the following fields: hostnames, bits, exponent, modulus, comment. The fields are separated by spaces.

Hostnames is a comma-separated list of patterns (\* and ? act as wildcards). Each pattern is matched against the fully-qualified host names (when authenticating a client) or against the user-supplied name (when authenticating a server). A pattern may be preceded by '!' to indicate negation; if the hostname matches a negated pattern, it is not accepted (by that line) even if it matched another pattern on the line.

Bits, exponent, and modulus are taken directly from the host key. They can be obtained from `MULTINET:SSH_HOST_KEY.PUB`. The optional comment field continues to the end of the line, and is not used. Lines starting with # and empty lines are ignored as comments. When performing host authentication, authentication is accepted if any matching line has the proper key.

It is permissible (but not recommended) to have several lines or different host keys for the same names. This happens when short forms of host names from different domains are put in the file. It is possible that the files contain conflicting information. Authentication is accepted if valid information can be found from either file.

**Note:** The lines in these files are hundreds of characters long. Instead of typing in the host keys, generate them by a script or by copying `MULTINET:SSH_HOST_KEY.PUB` and adding the host names at the front.

### Example

```
bos,bos.example.com,...,10.0.0.41
1024 37 159...93 bos.example.com
```

## SSH Logicals

These logicals are used with the SSH server in the system logical name table.

### SSH\_DIR

Points to the directory where the SSH1 configuration, master server log file, and host key files are kept. Normally, this is `MULTINET_SPECIFIC_ROOT:[MULTINET]`. It is defined in `START_SSH.COM`.

### **SSH\_EXE**

Points to the directory where SSH executables are kept. Normally, this is `MULTINET_COMMON_ROOT: [MULTINET]`. It is defined in `START_SSH.COM`.

### **SSH\_LOG**

Points to the directory where the log files are kept. Normally, this is `MULTINET_SPECIFIC_ROOT: [MULTINET.LOG]`. It is defined in `START_SSH.COM`.

### **SSH\_TERM\_MBX**

Mailbox used by `SSHD_MASTER` to receive termination messages from `SSHD` daemon processes. **Do not change this logical name.** This is created by the `SSHD_MASTER` process.

### **MULTINET\_SSH\_ACC\_REJ\_LOG\_FILE**

If the user has set a log file to log connection accept and reject messages, this logical will be defined and will provide the name of the log file. This logical is set by using the `SET LOG-FILE` keyword in `MULTINET CONFIGURE/SERVER`, and should not be modified directly by the user.

### **MULTINET\_SSH\_ALLOW\_EXPIRED\_PW**

Allows logging in to an account when the account's password has expired due to `pwdlifetime` elapsing. This applies to all users and circumvents normal VMS expired-password checking, and therefore should be used with caution. An entry is made into the `SSH_LOG:SSHD.LOG` file when access is allowed using this logical name.

When access is allowed by way of this logical, the logical name table `LNMS$SSH_LOGICALS` contains a logical name constructed as `MULTINET_SSH_pid_PWDEXP` (where `pid` is the PID for the process). The system manager can use this to execute, for example, the `DCL SET PASSWORD` command in the site `SYLOGIN.COM` file.

### **MULTINET\_SSH\_ALLOW\_PREEXPIRED\_PW**

Allows logging in to an account when the password has been pre-expired. This applies to all users and circumvents normal VMS expired-password checking, and therefore should be used with caution. An entry is made into the `SSH_LOG:SSHD.LOG` file when access is allowed using this logical name.

When access is allowed by way of this logical, the logical name table `LNMS$SSH_LOGICALS` contains a logical name constructed as `MULTINET_SSH_pid_PWDEXP` (where `pid` is the PID for the process).

The system manager can use this to execute, for example, the `DCL SET PASSWORD` command in the site `SYLOGIN.COM` file.

#### **MULTINET\_SSH\_DISPLAY\_SYS\$ANNOUNCE**

The SSH v1 protocol does not allow for the display of `SYS$ANNOUNCE` prior to logging in. If this logical is set, the contents of `SYS$ANNOUNCE` is displayed immediately after successful authentication and prior to the display of the contents of `SYS$WELCOME`.

#### **MULTINET\_SSH\_ENABLE\_SSH1\_CONNECTIONS**

Set by the MultiNet master server process to enable SSH V1 sessions.

#### **MULTINET\_SSH\_KEYGEN\_MIN\_PW\_LEN**

Defines the minimum passphrase length when one is to be set in `SSHKEYGEN`. If not defined, defaults to zero.

#### **MULTINET\_SSH\_LOG\_ACCEPTS**

When set, causes the server to log successful connection requests as either an `OPCOM` message or a line in a log file. Specified by the `SET LOG-ACCEPT` command in `MULTINET CONFIGURE/SERVER`. Note that the server does not use the information set in the `ACCEPT-HOSTS` keyword in `CONFIGURE/SERVER`. Rather, it uses the `AllowHosts` and `DenyHosts` keywords in the SSH server configuration file. Also, a successful connection request doesn't equate to a successful authentication request. This logical should not be modified directly by the user.

#### **MULTINET\_SSH\_LOG\_MBX**

Points to the OpenVMS mailbox used to log connection accept and reject messages. This must not be modified by the user.

#### **MULTINET\_SSH\_LOG\_REJECTS**

When set, causes the server to log rejected connection requests as either an `OPCOM` message or a line in a log file. Specified by the `SET LOG-REJECT` command in `MULTINET CONFIGURE/SERVER`. Note that the server does not use the information set in the `REJECT-HOSTS` keyword in `CONFIGURE/SERVER`. Rather, it uses the `AllowHosts` and `DenyHosts` keywords in the SSH server configuration file. This logical should not be modified directly by the user.

**MULTINET\_SSH\_MAX\_SESSIONS**

Set this to the maximum number of concurrent SSH sessions you want to allow on the server system. If `MULTINET_SSH_MAX_SESSIONS` is not defined, the default is 1000. Setting `MULTINET_SSH_MAX_SESSIONS` to zero (0) causes an error. The value must be between 1 and 1000. The suggested place to set this is in `START_SSH.COM`. You must restart SSH for these changes to take effect.

**MULTINET\_SSH\_PARAMETERS\_n**

These values are set by MultiNet and must not be modified by the user.

**MULTINET\_SSH\_USE\_SYSGEN\_LGI**

If defined, causes SSHD to use the VMS `SYSGEN` value of `LGI_PWD_TMO` to set the login grace time, overriding anything specified in the command line or the configuration file.

# 28. Configuring the Secure Shell (SSH) 2 Server

This chapter describes how to configure and maintain the MultiNet Secure Shell (SSH) server v2.

This is the server side of the software that allows secure interactive connections to other computers in the manner of rlogin/rshell/telnet. The SSH server has been developed to discriminate between SSH v1 and SSH v2 protocols, so the two protocols can coexist simultaneously on the same system.

## SSH1 and SSH2 Differences

SSH1 and SSH2 are different, and incompatible, protocols. The MultiNet SSH1 implementation is based on the version 1.5 protocol, and the SSH2 implementation is based on the V2 protocol. While SSH2 is generally regarded to be more secure than SSH1, both protocols are offered by MultiNet, and although they are incompatible, they may exist simultaneously on a MultiNet system. The MultiNet server front-end identifies what protocol a client desires to use, and will create an appropriate server for that client.

The cryptographic library used by MultiNet SSH2 is FIPS 140-2 level 2 compliant, as determined by the Computer Security Division of the National Institute of Science and Technology (NIST).

**Note:** You must install the DEC C 6.0 backport library on all OpenVMS VAX v6.0 and earlier systems prior to using SSH. This is the AACRT060.A file. You can find the ECO on the MultiNet CD in the following directory: VAX55\_DECC\_RTL.DIR.

## Restrictions:

When using SSH2 to connect to a VMS server, if the VMS account is set up with a secondary password, SSH2 does not prompt the user for the secondary password. If the VMS primary password entered is valid, the user is logged in, bypassing the secondary password.

When using SSH2 to execute single commands (in the same manner as RSHHELL), some keystrokes like CTRL+Y are ignored. In addition, some interactive programs such as HELP may not function as expected. This is a restriction of SSH2. If this behavior poses a problem, log into the remote system using SSH2 in interactive mode to execute the program.

# Understanding the MultiNet Secure Shell Server

Secure Shell daemon (SSHD) is the daemon program for SSH2 that listens for connections from clients. The server program replaces rshell and telnet programs. The server/client programs provide secure encrypted communications between two untrusted hosts over an insecure network. A new daemon is created for each incoming connection. These daemons handle key exchange, encryption, authentication, command execution, and data exchange.

## Servers and Clients

A MultiNet SSH server is an OpenVMS system that acts as a host for executing interactive commands or for conducting an interactive session. The server software consists of two pieces of software (for future reference, “SSHD” will refer to both SSHD\_MASTER and SSHD, unless otherwise specified):

- SSHD\_MASTER, recognizes the differences between SSH v1 and SSH v2 and starts the appropriate server. If the request is for SSH v1, then the existing SSH v1 server is run; if the request is for SSH v2, then the SSH v2 server is run.
- SSHD, a copy of which is spawned for each connection instance. SSHD handles all the interaction with the SSH client.

A client is any system that accesses the server. A client program (SSH) is provided with MultiNet, but any SSH client that uses SSH version 2 protocol may be used to access the server. Examples of such programs are MultiNet SSH, TCPware SSH, puTTY, SecureCRT, and other SSH programs on UNIX-based systems.

Each host has a key using DSA encryption and is usually 1024 bits long (although, the user may create a different-sized key, if desired). The same key may be used on multiple machines. For example, each machine in a VMScluster could use the same key.

When a client connects to the SSHD daemon:

- The client and server together, using the Diffie-Hellman key-exchange method, determine a 256-bit random number to use as the "session key". This key is used to encrypt all further communications in the session.
- Note that this key may be renegotiated between the client and the server on a periodic basis by including the `RekeyIntervalSeconds` keyword in the server configuration file (`SSH2_DIR:SSHD2_CONFIG`). This is desirable because during long sessions, the more data that is exchanged using the same encryption key, the more likely it is that an attacker who is watching the encrypted traffic could deduce the session key.
- The server informs the client which encryption methods it supports. See the description of the `CIPHERS` configuration keyword for the encryption methods supported.
- The client selects the encryption algorithm from those offered by the server.
- The client and the server then enter a user authentication dialog. The server informs the client which authentication methods it supports, and the client then attempts to authenticate the user by using some or all of the authentication methods.

The following authentication algorithms are supported:

- public-key (DSA keys)
- host-based
- password keyboard-interactive
- Kerberos V5 (password, kerberos-tgt, kerberos-1, kerberos-tgt-1, kerberos-2, kerberos-tgt-2)
- Certificate

System security is not improved unless the `RLOGIN` and `RSHELL` services are disabled.

If the client authenticates itself successfully, a dialog is entered for preparing the session. At this time the client may request things like:

- forwarding X11 connections
- forwarding TCP/IP connections
- forwarding the authentication agent connection over the secure channel

Finally, the client either requests an interactive session or execution of a command. The client and the server enter session mode. In this mode, either the client or the server may send data at any time, and such data is forwarded to/from the virtual terminal or command on the server side, and the user terminal in the client side. When the user program terminates and all forwarded X11 and other connections have been closed, the server sends command exit status to the client, and both sides exit.



# Expired Password Handling

The SSH2 server supports expired password changing for interactive accounts without the `CAPTIVE` or `RESTRICTED` flags set and, via the `DCL SET PASSWORD` command. When an expired password is detected, the server will behave as if a `SET PASSWORD` command was specified by the user as a remotely-executed command (e.g., `$ ssh foo set password`), and the user will be logged out after changing the password. The user may then log in again using the changed password.

For `CAPTIVE` or `RESTRICTED` accounts, or for those accounts where `LGICMD` is set in the `UAF` record, the scenario is different. In these cases, the server can't directly execute `SET PASSWORD` command, because the command procedure specified in the `LGICMD` field of the `UAF` record will override the SSH server attempting to do a `SET PASSWORD` command. For these types of accounts, the system manager and/or user can use the value of the `LOGIN_FLAGS` for the process (normal interactive sessions may also examine these flags). For SSH logins, these flags will reflect:

- new mail has been received (`JPI$M_NEW_MAIL_AT_LOGIN`)
- the password is about to expire (`JPI$M_PASSWORD_WARNING`)
- the password has expired (`JPI$M_PASSWORD_EXPIRED`)

The `DCL` lexical function `F$GETJPI` may be used to examine these flags, as can the `$GETJPI (W)` system service or `LIB$GETJPI RTL` function. When an expired password value is detected, the user may then execute a `SET PASSWORD` command in the command procedure run for the account.

For example:

```
$!  
$! Login_flags:  
$!   1 = new mail messages waiting (JPI$M_NEW_MAIL_AT_LOGIN)  
$!   4 = password expired during login (JPI$M_PASSWORD_EXPIRED)  
$!   5 = password expires within 5 days (JPI$M_PASSWORD_WARNING)  
$!  
$ flags = f$getjpi("", "LOGIN_FLAGS")  
$ new_flags = (flags/2)*2  
$ if new_flags .ne. flags then write sys$output "New mail waiting"  
$!  
$! Note - new_flags is used below because it has the NEW_MAIL_AT_LOGIN$  
$! bit stripped. The rest of the possible values are all  
$! discrete; i.e., you can't have combinations of them at the  
$! same time.  
$!  
$ if new_flags .eq. 4 then write sys$output "Password expired during login"  
$ if new_flags .eq. 5 then write sys$output "Password expires within 5 days"  
$!
```

When an account in the SYSUAF has an expired password and the system `syslogin.com` or user's `login.com` has a `SET TERM` command, a warning message will be displayed prior to prompting to change the password as shown in the following example:

```
Your password has expired; you must set a new password to log in

% SET-W-NOTSET, error modifying DKA0:
-SET-E-INVDEV, device is invalid for requested operation

Old password:
```

The way to suppress these warning messages would be to check for the appropriate login flag, ignoring any `SET TERM` commands. For example:

```
$ flags = $getjpi("", "LOGIN_FLAGS")
$ new_flags = (flags/2)*2
$ if new_flags.eq.4 then goto skip_the_inquiry
```

## Break-In and Intrusion Detection

Care must be exercised when configuring the SSH clients and server to minimize problems due to intrusion records created by OpenVMS security auditing. The SSH user should consult the system manager to determine the authentication methods offered by the SSH server. The client should then be configured to not attempt any authentication method that is not offered by the server.

If a client attempts authentication methods not offered by the server, the OpenVMS security auditing system may log several intrusion records for each attempt to create a session to that server. The result being that the user could be locked out and prevented from accessing the server system without intervention from the server's system manager.

The authentication methods to be offered by the server are determined by the configuration keywords `AllowedAuthentications` and `RequiredAuthentications`. The number of intrusion records to be logged for any attempted SSH session is determined by the `StrictIntrusionLogging` configuration keyword.

When `StrictIntrusionLogging` is set to `YES` (the default), each method that is tried and fails causes an intrusion record to be logged. The following rules apply:

- When `HostBased` or `PublicKey` authentications are attempted and fail, one intrusion record is logged for each failed method.
- When password authentication is attempted, one intrusion record is logged for each failed password.

### Example 1:

The server is set up to allow `HostBased` and `password` authentication; also, up to three password attempts are allowed. If all methods fail, four intrusion records are logged:

1 for the failed `HostBased`

3 for the failed `password` attempts, one per attempt

When `StrictIntrusionLogging` is set to `NO`, it has the effect of relaxing the number of intrusions logged. Overall failure of all authentication methods simply counts as a single failure, except for `password` authentication. The following rules apply:

- When `password` authentication is attempted, one intrusion record is logged for each failed `password`.
- When any of `HostBased` or `PublicKey` authentication fails, and `password` authentication is not attempted, exactly one intrusion record is logged, as opposed to one for each failed method.
- When any of `HostBased` or `PublicKey` authentication fails, but `password` authentication is attempted and succeeds, the only intrusion record(s) logged is one for each failed `password` attempt.

### Example 2:

The server is set up to allow `HostBased` and `password` authentication; also, up to three password attempts are allowed. If all methods fail, three intrusion records are logged:

0 for the failed `HostBased`

3 for the failed `password` attempts, one per attempt

### Example 3:

The server is set up to allow `HostBased` and `password` authentication; also, up to three password attempts are allowed. `HostBased` and `RSA` fail, but `password` authentication is successful after 1 failed `password`. Therefore, one intrusion record is logged:

0 for the failed `HostBased`

1 for the failed `password` attempt

### Example 4:

The server is set up to allow `HostBased` and `PublicKey` authentication, but not `password` authentication. If all methods fail, one intrusion record is logged.

## Example 5:

The server is set up to allow `HostBased` and `PublicKey` authentication, but not password authentication. `HostBased` authentication fails, but `PublicKey` succeeds. No intrusion records are logged.

# Configuring SSHD Master

SSHD Master is configured using the `MULTINET CONFIGURE/SERVER` command, selecting SSH, and using the following parameters:

**Note:** The only supported methods to start SSHD Master are to restart the MultiNet server if SSH is not currently running, or to use the `MULTINET NETCONTROL SSH START` command. All of these options are set using `MULTINET CONFIGURE/SERVER`, and modifying the SSH service.

### **bits *n***

Specifies the number of bits in the server key. The default is 768.

### **debug *debug-level***

Disables or enables debugging levels. Values are between 0 and 50. Zero (0) disables debugging and higher values turn on successively more debugging.

### **ipv4-disable**

Disables the server from listening on an IPv4 socket.

### **ipv6-disable**

Disables the server from listening on an IPv6 socket.

### **enable-ssh2**

Enables SSH V2 sessions.

**listen-address**

Specify the IPV4 or IPV6 address on which to listen for connect request. This may be a valid IPV4 or IPV6 address, or ANY to listen on all addresses. If not specified, the default is to listen on all IPV4 and IPV6 addresses.

**port *n***

Specifies the port on which the server listens for connections. The default is 22.

**quiet\_mode**

Specifies that nothing is sent to the system log. Normally, the beginning, authentication, and termination of each connection is logged.

**ssh2-config-file *filename***

Specifies the name of the configuration file. The default is SSH2\_DIR:SSHD2\_CONFIG.

**verbose**

Specifies that verbose message logging will be performed by SSHD Master.

## SSH2 Configuration File

SSHD reads configuration data from its configuration file. By default, this file is SSH2\_DIR:SSHD2\_CONFIG. The file contains keyword value pairs, one per line. Lines starting with # and empty lines are interpreted as comments. The following keywords are possible. Keywords are case insensitive.

Keyword	Value	Default	Description
AllowedAuthentications	List	Publickey, Password	Permitted techniques. Valid values are:  Keyboard-interactive, password, public-key,

			<p>hostbased, kerberos-1, kerberos-tgt-1, kerberos-2, kerberos-tgt-2</p> <p>Along with RequiredAuthentications, the system administrator can force the users to complete several authentications before they are considered authenticated.</p>
AllowedPasswordAuthentications	List	kerberos, local	<p>Specifies the different password authentication schemes that are allowed.</p> <p>Only kerberos and local are acceptable.</p>
AllowGroups	List		Access control by UAF rights list entries
AllowHosts	Host list		Access control by hostname
AllowShosts	Host list		Access control by hostname
AllowTcpForwarding	Y/N	Y	Enable TCP port forwarding
AllowTcpForwardingForUsers	User list		Per-User forwarding
AllowTcpForwardingForGroups	Rights list		Per-Rights list ID forwarding
AllowUsers	User list		Access control by username
AllowX11Forwarding	Y/N	Y	Enable X11 forwarding

AuthInteractiveFailureTime out	Seconds	2	Delay, in seconds, that the server delays after a failed attempt to log in using keyboard-interactive and password authentication.
AuthKbdInt.NumOptional	Number	0	Specifies how many optional submethods must be passed before the authentication is considered a success. (Note that all reported submethods must always be passed.) See <code>AuthKbdInt.Optional</code> for specifying optional submethods, and <code>AuthKbdInt.Required</code> for required submethods. The default is 0, although if no required submethods are specified, the client must always pass at least one optional submethod.
AuthKbdint.Optional	List	None	Specifies the optional submethods keyboard-interactive will use. Currently only the submethod password is defined.  <code>AuthKbdInt.NumOptional</code> specifies how many optional submethods must be passed. The keyboard-interactive authentication method is considered a success when the specified amount of optional submethods and all required submethods are passed.

AuthKbdInt.Required			Specifies the required submethods that must be passed before the keyboard-interactive authentication method can succeed.
AuthKbdInt.Retries	Number	3	Specifies how many times the user can retry keyboard-interactive.
AuthorizationFile	Filename	Authorization	Authorization file for publickey authentication.
AuthPublicKey.MaxSize	Number	0	Specifies the maximum size of a publickey that can be used to log in. Value of 0 disables the check.
AuthPublicKey.MinSize	Number	0	Specifies the minimum size of a publickey that can be used to log in. Value of 0.
Cert.RSA.Compat.HashScheme	md5 or sha	md5	Previous clients and servers may use hashes in RSA certificates incoherently (sometimes SHA-1 and sometimes MD5). This specifies the hash used when a signature is sent to old versions during the initial key exchanges.
BannerMessageFile	Filename	SYS\$ANNOUNCE	Message sent to the client before authentication begins.
CheckMail	Y/N	Y	Display information about new mail messages when logging in



Ciphers	Cipher list		Encryption ciphers offered
DenyGroups	Rights list		Deny access for UAF rightslist identifiers
DenyHosts	Host list		Deny access for hosts
DenySHosts	Host list		Deny access for hosts
DenyTcpForwardingForUsers	User list		Forbid forwarding for listed users
DenyTcpForwardingForGroups	Rights list		Forbid forwarding for listed rightslist names
DenyUsers	User list		Access control by username
FascistLogging	Y/N	Y	Verbose logging
ForwardACL	Pattern	None	With this option, you can have more fine-grained control over what the client is allowed to forward, and to where. See <i>“ForwardACL Notes”</i> below.
ForwardAgent	Y/N	Y	Enable agent forwarding
HostCA	Certificate	None	Specifies the CA certificate (in binary or PEM (base64) format) to be used when authenticating remote hosts. The certificate received from the host must be issued by the specified CA and must contain a correct alternate name of type DNS (FQDN). If no CA certificates are specified in the configuration file, the protocol tries to do key exchange with

			ordinary public keys. Otherwise, certificates are preferred. Multiple CAs are permitted.
HostCANoCRLs	Certificate	None	Similar to HostCA, but disables CRL checking for the given CA certificate.
HostCertificateFile	Filename	None	This keyword works very much like PublicHostKeyFile, except that the file is assumed to contain an X.509 certificate in binary format. The keyword must be paired with a corresponding HostKeyFile option. If multiple certificates with the same public key type (DSS or RSA) are specified, only the first one is used.
HostbasedAuthForceClient HostnamedNSMatch	Y/N	N	Host name given by client.
Hostkeyfile	Filename	Hostkey	Host key filename
HostSpecificConfig	Pattern	None	Specifies a subconfiguration file for this server, based on the hostname of the client system.
IdentityFile	Filename	Identification	Identity filename
IdleTimeout	Time	0 = none	Set idle timeout (in seconds)

IgnoreRhosts	Y/N	N	Don't use rhosts and shosts for hostbased authentication for all users
IgnoreRootRhosts	Y/N	Y	Don't use rhosts and shosts files for authentication of SYSTEM
KeepAlive	Y/N	Y	Send keepalives
LdapServers	Server URL	None	<p>Specified as  <code>ldap://server.domain-name:389</code></p> <p>CRLs are automatically retrieved from the CRL distribution point defined in the certificate to be checked if the point exists. Otherwise, the comma-separated server list given by option <code>LdapServers</code> is used. If intermediate CA certificates are needed in certificate validity checking, this option must be used or retrieving the certificates will fail.</p>
ListenAddress	IP address	0.0.0.0	Listen on given interface
Macs	Algorithm		Select MAC (Message Authentication Code) algorithm
MapFile	Filename	None	This keyword specifies a mapping file for the preceding <code>Pki</code> keyword. Multiple mapping files are permitted per one <code>Pki</code> keyword. The

			mapping file format is described below.
MaxBroadcastsPerSecond	#broadcasts	0	Listen for UDP broadcasts
NoDelay	Y/N	N	Enable Nagel Algorithm
PasswordAuthentication	Y/N	Y	Permit password authentication
PasswordGuesses	#guesses	3	Limit number of password tries to specified number
PermitEmptyPasswords	Y/N	N	Permit empty (blank) passwords
PermitRootLogin	Y/N	N	SYSTEM can log in
Pki	Filename	None	This keyword enables user authentication using certificates. CA-certificate must be an X.509 certificate in binary format. This keyword must be followed by one or more MapFile keywords. The validity of a received certificate is checked separately using each of the defined Pki keywords in turn until they are exhausted (in which case the authentication fails), or a positive result is achieved. If the certificate is valid, the mapping files are examined to determine whether the certificate allows the user to log in. A correct signature generated by a

			matching private key is always required.
PkiDisableCrls	Y/N	Y	This keyword disables CRL checking for the Pki keyword, if argument is Y.
PrintMotd	Y/N	Y	Display SYS\$WELCOME when logging in
PublicHostKeyFile	Filename	Hostkey.pub	Host key file location
QuietMode	Y/N	N	Quiet mode
RandomSeedFile	Filename	Random_seed	Random seed file
RekeyIntervalSeconds	# seconds	0	Frequency of rekeying
RequiredAuthentication	Authentication list		Authentications client must support
RequireReverseMapping	Y/N	N	Remote IP address must map to hostname
ResolveClientHostName	Y/N	Y	Controls whether the server will try to resolve the client IP address at all, or not. This is useful when you know that the DNS cannot be reached, and the query would cause additional delay in logging in. Note that if you set this to N, you should not set RequireReverseMapping to Y.
RSAAuthentication	Y/N	Y	Enable RSA authentication

SendKeyGuess	Y/N	Y	This parameter controls whether the server will try to guess connection parameters during key exchange, or not. Some clients do not support key exchange guesses and may fail when they are present.
SftpSysLogFacility	log facility	None	Defines the log facility the SFTP server will use
StrictIntrusionLogging	Y/N	Y	Determine how intrusion records are created by failed authentication attempts.
StrictModes	Y/N	N	Strict checking for directory and file protection.
SyslogFacility	Facility	AUTH	Defines what log facility to be used when logging server messages.
Terminal.AllowUsers	pattern	All users	List users that are allowed terminal (interactive) access to the server.
Terminal.DenyUsers	pattern	None	List users that are denied terminal (interactive) access to the server.
Terminal.AllowGroups	pattern	All groups	Similar to Terminal.AllowUsers but matches groups instead of usernames.
Terminal.DenyGroups	pattern	None	Similar to Terminal.DenyUsers but

			matches groups instead of usernames
UserConfigDirectory	Directory	SYS\$LOGIN:	Location of user SSH2 directories
UserKnownHosts	Y/N	Y	Respect user [.ssh2] known hosts keys
UserSpecificConfig	Pattern	None	Specifies a subconfiguration file for this server, based on user logging in.
VerboseMode	Y/N	N	Verbose mode

The keywords /MAC and /CIPHER have discrete values, plus there are values that actually denote a grouping of 2 or more of the discrete values. Each of these values may be put in the configuration file (SSH2\_DIR:SSHD2\_CONFIG).

<b>MACs</b>	Discrete values: hmac-sha1, hmac-sha256, hmac-md5, hmac-ripemd160, none
	Group ANYMAC consists of: hmac-sha1, hmac-sha256, hmac-md5, hmac-ripemd160
	Group ANY consists of: hmac-sha1, hmac-sha256, hmac-md5, hmac-ripemd160, none
	Group ANYSTD consists of: hmac-sha1, hmac-md5, none
	Group ANYSTDMAC consists of: hmac-sha1, hmac-md5
<b>Ciphers</b>	Discrete values: 3des, aes, blowfish, aes128-ctr, aes128-cbc, aes192-ctr, aes192-cbc, aes256-ctr, aes256-cbc, 3des-ctr, 3des-cbc, blowfish-ctr, blowfish-cbc, des-cbc, rc2-cbc, none

<p><b>Group ANYSTDCIPHER consists of:</b></p> <p>aes128-ctr, aes128-cbc, aes192-ctr, aes192-cbc, aes256-ctr, aes256-cbc, 3des-ctr, 3des-cbc, blowfish-ctr, blowfish-cbc</p>
<p><b>Group ANY consists of:</b></p> <p>aes128-ctr, aes128-cbc, aes192-ctr, aes192-cbc, aes256-ctr, aes256-cbc, 3des-ctr, 3des-cbc, blowfish-ctr, blowfish-cbc, des-cbc, rc2-cbc, none</p>
<p><b>Group ANYCIPHER consists of:</b></p> <p>aes128-cbc, 3des-cbc, twofish128-cbc, cast128-cbc, twofish-cbc, blowfish-cbc, aes192-cbc, aes256-cbc, twofish192-cbc, twofish256-cbc, arcfour, des-cbc, rc2-cbc</p>
<p><b>Group ANYSTD consists of:</b></p> <p>aes128-cbc, 3des-cbc, twofish128-cbc, cast128-cbc, twofish-cbc, blowfish-cbc, aes192-cbc, aes256-cbc, twofish192-cbc, twofish256-cbc, arcfour, none</p>

A discrete value or a group identifier may be used with MACS and CIPHERS. For example, in the configuration file, the following examples could be used:

Ciphers ANYCIPHER

Ciphers 3des, aes128-cbc

MACs ANYMAC

MACs hmac-sha1

Aliases may be used for some standard ciphers:

Alias	Value
aes	aes128-cbc
3des	3des-cbc
blowfish	blowfish-cbc



## HostSpecificConfig Notes:

The global server file (`SSH_DIR:SSHD2_CONFIG`) now can use the keyword `HostSpecificConfig` to allow the specification of a configuration file based on the client system. These lines are specified as:

```
HostSpecificConfig      hostname      subconfig-file
```

`hostname` will be used to match the client host, as specified under option `AllowHosts`. The file `subconfig-file` will then be read, and configuration data amended accordingly. The file is read before any actual protocol transactions begin, and you can specify most of the options allowed in the main configuration file. You can specify more than one subconfiguration file, in which case the patterns are matched and the files read in the order specified. Later defined values of configuration options will either override or amend the previous value, depending on which option it is. The effect of redefining an option is described in the documentation for that option. For example, setting `Ciphers` in the subconfiguration file will override the old value, but setting `AllowUsers` will amend the value.

The `subconfig-file` will be assumed by default to exist in the `SSH2_DIR` directory. However, this may be overridden by specifying a complete directory/file specification. For example:

```
HostSpecificConfig      bos.example.com      dka0:[sshconfigs]bosconfig.dat
HostSpecificConfig      clt.example.com      cltconfig.dat
```

In the first instance, an incoming connection from `bos.example.com` will use the subconfig file `dka0:[sshconfigs]bosconfig.dat`. In the second example, an incoming connection from `clt.example.com` will use `ssh2_dir:cltconfig.dat`.

Unlike `ssh2_config`, the subconfig files may have configuration blocks, or stanzas, in them. They are used per-host. The subconfiguration heading is interpreted identically to what is described above (i.e. with `UserSpecificConfig`, the pattern is of the format “hostname”).

**Note:** If the subconfig file cannot be found or cannot be parsed successfully for any reason, access to the system will be denied for the system to which the subconfig file applies.

## UserSpecificConfig Notes:

The global server file (`SSH2_DIR:SSHD2_CONFIG`) can use the keyword `UserSpecificConfig` to allow the specification of a configuration file based on the username of the user who’s logging into the server. These keywords are of the form:

```
UserSpecificConfig user[%group] [@host] subconfig-file
```

*pattern* will be used to match the username, as specified under the option `AllowUsers`. The file *subconfig-file* will then be read, and configuration data amended accordingly. The file is read before any actual protocol transactions begin, and you can specify most of the options allowed in the main configuration file. You can specify more than one subconfiguration file, in which case the patterns are matched and the files read in the order specified. Later defined values of configuration options will either override or amend the previous value, depending on which option it is. The effect of redefining an option is described in the documentation for that option. For example, setting `Ciphers` in the subconfiguration file will override the old value, but setting `AllowUsers` will amend the value.

Unlike `sshd2_config`, the subconfig files may have configuration blocks, or stanzas, in them. They are used per user. The subconfiguration heading is interpreted identically to what is described above (i.e., with `UserSpecificConfig`, the pattern is of the format *user[%group] [@host]*).

The *subconfig-file* will be assumed by default to exist in the `SSH2_DIR` directory. However, this may be overridden by specifying a complete directory/file specification. For example:

```
UserSpecificConfig dilbert dka0:[sshconfigs]dilbert.dat
UserSpecificConfig boss@lima.beans.com pointyhair.dat
```

In the first instance, an incoming connection for user `alice` will use the subconfig file

`dka0:[sshconfigs]alice.dat`. In the second example, an incoming connection from user `bob` at system `lima.beans.com` will use `ssh2_dir:bob.dat`.

**Note:** If the subconfig file cannot be found or cannot be parsed successfully for any reason, access to the system will be denied for the user to which the subconfig file applies.

## KEYBOARD-INTERACTIVE Notes:

At this point, `KEYBOARD-INTERACTIVE` mode is simply another form of password authentication. The user won't notice anything different with this mode. In the future, Process Software may implement items such as system passwords, secondary passwords, and true VMS-style password changing using this authentication method. As other clients support the use of the `KEYBOARD-INTERACTIVE` authentication method for doing password authentication (without using any external callouts from the mechanism such as SecureID cards), the server should support those clients.

## ForwardACL Notes

With this option, you can have more fine-grained control over what the client is allowed to forward, and to where. Format for this option is:

```
[allow|deny] [local|remote] user-pat forward-pat [originator-pat]
```

*user-pat* will be used to match the client-user, as specified under the option

`UserSpecificConfig.forward-pat` is a pattern of format *host-id[%port]*. This has different interpretations, depending on whether the ACL is specified for local or remote forwards. For local forwards, the *host-id* will match with the target host of the forwarding, as specified under the option `AllowHosts.port` will match with the target port. Also, if the client sent a host name, the IP address will be looked up from the DNS, which will be used to match the pattern. For remote forwardings, where the forward target is not known (the client handles that end of the connection); this will be used to match with the listen address specified by the user (and as such is not as usable as with local forwards). *port* will match the port the server is supposed to be listening to with this forward. With local forwards, *originator-pat* will match with the originator address that the client has reported. Remember, if you do not administer the client machine, users on that machine may use a modified copy of SSH that can be used to lie about the originator address. Also, with NATs (Network Address Translation), the originator address will not be meaningful (it will probably be an internal network address). Therefore, you should not rely on the originator address with local forwards, unless you know exactly what you are doing. With remote forwards, *originator-pat* will match with the IP address of the host connecting to the forwarded port. This will be valid information, as it is the server that is checking that information.

If you specify any allow directives, all forwards in that class (local or remote) not specifically allowed will be denied (note that local and remote forwards are separate in this respect, e.g., if you have one “allow remote” definition, local forwards are still allowed, pending other restrictions). If a forward matches with both allow and deny directives, the forwarding will be denied. Also, if you have specified any of the options `[Allow.Deny]TcpForwardingForUsers.Groups` or `AllowTcpForwarding`, and the forwarding for the user is disabled with those, an allow directive will not re-enable the forwarding for the user. Forwarding is enabled by default.

## MappingFileFormat

When certificates are used in user authentication, one or more mapping files determine whether the user can log to an account with a certificate. The mapping file must contain one or more lines in the following format:

```
account-id keyword arguments
```

Keyword must be one of the following: `Email`, `EmailRegex`, `Subject`, `SerialAndIssuer`, or `SubjectRegex`.

Arguments are different for each keyword. The following list describes each variation:

**Email**

arguments: an email address in standard format. If the certificate contains the email address as an alternate name, it is good for logging in as user *account-id*.

**Subject**

arguments: a subject name in DN notation (LDAP style). If the name matches the one in the certificate, the certificate is good for logging in as user *account-id*.

**SerialAndIssuer**

arguments: a number and an issuer name in DN notation (LDAP style), separated by whitespace. If the issuer name and serial number match those in the certificate, the certificate is good for logging in as user *account-id*.

**EmailRegex**

arguments: a regular expression (*egrep syntax*). If it matches an `alternate` (of type `email-address`) in the certificate, the certificate is good for logging in as user `account-id`. As a special feature, if `account-id` contains a string `%subst%`, it is replaced by the first parenthesized substring of the regular expression before comparing it with the account the user is trying to log into.

**SubjectRegex**

Works identically to `EmailRegex`, except it matches the regular expression to the canonical subject name in the received certificate.

Empty lines and lines beginning with `#` are ignored.

**EXAMPLE: MAPPINGFILE**

```
guest email guest@domain.org
guest subject C=Fl,O=Company Ltd., CN=Guest User
guest SerialAndUser 123 C=Fl, O=Foo\Ltd., CN=Test CA
%subst% EmailRegex ([a-z]+)@domain.\org
%subst% Subjectregex ^C=Fl,O=Company,CN=([a-z]+)$
```

The example `EmailRegex` permits in users with email addresses with domain `domain.org` and usernames that contain only letters, each user to the account that corresponds to the username part of the email address.

The example `SubjectRegex` lets in all users with fields `C=F1` and `O=Company` in the subject name if their `CN` field contains only letters and is the account name they are trying to log into.

Note the `^` and `$` at the beginning and end of the regular expression; they are required to prevent the regular expression from matching less than the whole string (subject name).

Note also that all characters interpreted by the regular expression parser as special characters must be escaped with a backslash if they are a part of the subject name. This also means that the backslash in the `SerialAndIssuer` example would have to be escaped with another backslash if the same subject name was used in a `SubjectRegex` rule.

# Starting the SSH Server for the First Time

Follow these instructions for using SSH for the first time.

1. Use the `MULTINET CONFIGURE/SERVER` command to enable the SSH v2 server.

```
$ MULTINET CONFIGURE/SERVER
MultiNet Server Configuration Utility V5.6
[Reading in configuration from MULTINET:SERVICES.MASTER_SERVER]
SERVER-CONFIG>SHOW/FULL SSH
Service "SSH": ***DISABLED***
    INIT() = Merge_Image
    Program = "MULTINET:LOADABLE_SSH_CONTROL"
    Priority = 5
    Parameters = "enable-ssh1"
                "enable-ssh2"

SERVER-CONFIG>ENABLE SSH
SERVER-CONFIG>EXIT
[Writing configuration to
MULTINET_COMMON_ROOT:[MULTINET]SERVICES.MASTER_SERVER]
```

**Note:** The parameter `enable-ssh2` must be set. If it is not set, SSH v2 sessions will not be accepted by the server.

2. Use SSHKEYGEN /SSH2 to generate an SSH2 key and to create the server keys in the SSH2\_HOSTKEY\_DIR directory:

```
$ DEFINE MULTINET_SSH2_HOSTKEY_DIR -  
$ MULTINET_SPECIFIC_ROOT:[MULTINET.SSH2.HOSTKEYS]  
$ MULTINET_SSHKEYGEN /SSH2 /HOST  
Generating 1024-bit dsa key pair  
 8 .oOo.oOoo.oO  
Key generated.  
1024-bit dsa, lillies@sfo.example.com, Mon Aug 04 2019 09:19:47  
Private key saved to multinet_ssh2_hostkey_dir:hostkey.  
Public key saved to multinet_ssh2_hostkey_dir:hostkey.pub
```

3. Copy the template server configuration file to the ssh2\_dir: directory renaming it SSHD2\_CONFIG.:

```
$ COPY MULTINET_SPECIFIC_ROOT:[MULTINET.SSH2]SSHD2_CONFIG.TEMPLATE -  
$ MULTINET_SPECIFIC_ROOT:[MULTINET.SSH2]SSHD2_CONFIG.
```

4. Copy the template client configuration file to the ssh2\_dir: directory renaming it SSH2\_CONFIG.:

```
$ COPY MULTINET_SPECIFIC_ROOT:[MULTINET.SSH2]SSH2_CONFIG.TEMPLATE -  
$ MULTINET_SPECIFIC_ROOT:[MULTINET.SSH2]SSH2_CONFIG.
```

**Note:** As delivered, the template files provide a reasonably secure SSH environment. However, Process Software recommends these files be examined and modified appropriately to reflect the security policies of your organization.

5. Restart MultiNet. This creates the SSH server process and defines the SSH logical names.

```
$ @MULTINET:START_SERVER_RESTART  
$ SHOW=SYSTEM/PROCESS=3D "SSHD Master"  
7-JUL-2019 09:03:06.42  User: SYSTEM          Process ID:    00000057  
                        Node: PANTHR         Process name:  "SSHD Master"  
  
Terminal:  
User Identifier:      [SYSTEM]  
Base priority:        4  
Default file spec:    Not available  
Number of Kthreads:  1  
  
Devices allocated:   BG1:  
                    BG2:  
  
$ SHOW LOGICAL/SYSTEM *SSH*
```

```

"MULTINET_SSH2_HOSTKEY_DIR" =
    "MULTINET_SPECIFIC_ROOT:[MULTINET.SSH2.HOSTKEYS]"
"MULTINET_SSH2_KNOWNHOSTS_DIR" =
    "MULTINET_SPECIFIC_ROOT:[MULTINET.SSH2.KNOWNHOSTS]"
"MULTINET_SSH_ENABLE_SSH2_CONNECTIONS" = "1"
"SSH2_DIR" = "MULTINET_SPECIFIC_ROOT:[MULTINET.SSH2]"
"SSH_DIR" = "MULTINET_SPECIFIC_ROOT:[MULTINET]"
"SSH_EXE" = "MULTINET_COMMON_ROOT:[MULTINET]"
"SSH_LOG" = "MULTINET_SPECIFIC_ROOT:[MULTINET.SSH]"
"SSH_TERM_MBX" = "MBA36:"

```

# Configuring the SSH2 Server on a VMSccluster with a Common System Disk

When configuring the SSH2 server on a VMSccluster with a common system disk, you must create the appropriate directories on all cluster nodes other than the one on which MultiNet was originally installed. Note that this does not need to be done for cluster members that do not share a common system disk.

The following procedure should be followed on each cluster node other than the cluster node on which MultiNet was originally installed:

- Create the necessary directories:

```

$ CREATE/DIR MULTINET_SPECIFIC_ROOT:[MULTINET.SSH2]/PROT=(WO:RE,GR:RE)
$ CREATE/DIR
MULTINET_SPECIFIC_ROOT:[MULTINET:SSH2.KNOWNHOSTS]/PROT=(WO:R,GR:R)
$ CREATE/DIR
MULTINET_SPECIFIC_ROOT:[MULTINET.SSH2.HOSTKEYS]/PROT=(WO:R,GR:R)
$ CREATE/DIR MULTINET_SPECIFIC_ROOT:[MULTINET.SSH]/PROT=(WO:RE,GR:RE)

```

- Edit the MULTINET\_SPECIFIC\_ROOT:[MULTINET.SSH2] SSHD2\_CONFIG file as necessary. This may be copied from another cluster node, or it may be created fresh from the SSHD2\_CONFIG.TEMPLATE file.
- Edit the MULTINET\_SPECIFIC\_ROOT:[MULTINET.SSH2] SSH2\_CONFIG file as necessary. This may be copied from another cluster node, or it may be created fresh from the SSH2\_CONFIG.TEMPLATE file.
- Configure the SSH2 server using MULTINET CONFIGURE/SERVER
- Generate the SSH2 host keys using MULTINET SSHKEYGEN/SSH2/HOST

- (Re)start SSHD Master using `MULTINET NETCONTROL SSH RESTART`

# Changing SSH2 Configuration File After Enabling SSH2

If you make a change to the SSH configuration file after you have enable SSH, you must restart SSH for these changes to take effect.

```
$ MULTINET NETCONTROL SSH RESTART
```

**Note:** When issuing the `RESTART` command for SSH, all active SSH server sessions are terminated. Active client sessions are not affected.

## Connection and Login Process

To create a session, SSHD does the following:

1. `SSHD_MASTER` sees the connection attempt. It creates an `SSHD` process, passing the operating parameters to it. `SSHD` performs validation for the user.
2. Assuming the login is successful, `SSHD` creates a pseudo terminal for the user (an `FTAnn:` device). This device is owned by the user attempting to log in.
3. `SSHD` creates an interactive process on the pseudo terminal, using the username, priority, and privileges of the user who is attempting to log in. If a command was specified, it is executed and the session is terminated.
4. SSH generates the file `SSHD.LOG` for each connection to the SSH server. Many connections result in many log files. Instead of purging the files on a regular basis, use the following DCL command to limit the number of versions:

```
$ SET FILE /VERSION LIMIT=x MULTINET ROOT: [MULTINET.SSH] SSHD.LOG
```



**Note:** The value for `/VERSION_LIMIT` must not be smaller than the maximum number of simultaneous SSH sessions anticipated. If the value is smaller, SSH users may be prevented from establishing sessions with the server.

## FILES

### **MULTINET:HOSTS.EQUIV**

Contains host names, one per line. This file is used during `.rhosts` authentication. Users on those hosts are permitted to log in without a password, provided they have the same username on both machines. The hostname may also be followed by a username. Such users are permitted to log in as any user on the remote machine (except `SYSTEM`). Additionally, the syntax `+@group` can be used to specify netgroups. Negated entries start with dash (-). If the client host/user is matched in this file, login is permitted provided the client and server usernames are the same. Successful RSA host authentication is required. This file should be world-readable but writable only by `SYSTEM`.

It is never a good idea to use usernames in `hosts.equiv`. It means the named user(s) can log in as anybody, which includes accounts that own critical programs and directories. Using a username grants the user `SYSTEM` access. The only valid use for usernames is in negative entries.

### **MULTINET:SHOSTS.EQUIV**

Processed as `MULTINET:HOSTS.EQUIV`. May be useful in environments that want to run both `RSHELL/RLOGIN` and `SSH`.

### **MULTINET\_SSH2\_HOSTKEY\_DIR:HOSTKEY**

Contains the private part of the host key. This file does not exist when MultiNet is installed. The `SSH` server starts only with this file. This file must be created manually using the command:

```
$ MULTINET SSHKEYGEN /SSH2 /HOST.
```

This file should be owned by `SYSTEM`, readable only by `SYSTEM`, and not accessible to others.

To create a host key with a name that is different than what `SSHKEYGEN` creates, do one of the following:

- Generate with `SSHKEYGEN /SSH2/HOST` and simply rename the file(s).
- Generate without the `/HOST` switch and then name the file(s) whatever you want.

By default the logical name `SSH2_DIR` points to the `MULTINET_SPECIFIC_ROOT:[MULTINET.SSH2]` directory.

Refer to the *MultiNet User's Guide*, Chapter 8, for more details about `SSHKEYGEN`.

**MULTINET\_SSH2\_HOSTKEY\_DIR:HOSTKEY.PUB**

Contains the public part of the host key. This file should be world-readable but writable only by `SYSTEM`. Its contents should match the private part. This file is not used for anything; it is only provided for the convenience of the user so its contents can be copied to known hosts files.

**SSH2:SSH\_RANDOM\_SEED**

**SYS\$LOGIN:[.SSH]RANDOM\_SEED**

Contains a seed for the random number generator. This file should only be accessible by `SYSTEM`.

**SSH2\_DIR:SSHD2\_CONFIG**

Contains configuration data for the v2 `SSHD` server. This file should be writable by system only, but it is recommended (though not necessary) that it be world-readable.

**SYS\$LOGIN:[.SSH2].SHOSTS**

Permits access using `SSH2` only. For `SSH2`, this file is the same as for `.rhosts`. However, this file is not used by the `RLOGIN` and `RSHELL` daemon.

**SYS\$LOGIN:.RHOSTS**

This file contains host-username pairs, separated by a space, one per line. The given user on the corresponding host is permitted to log in without a password. The same file is used by `RLOGIN` and `RSHELL`. `SSH2` differs from `RLOGIN` and `RSHELL` in that it requires `RSA` host authentication in addition to validating the hostname retrieved from domain name servers (unless compiled with the `-with-rhosts` configuration option). The file must be writable only by the user. It is recommended that it not be accessible by others. It is possible to use `netgroups` in the file. Either host or username may be of the form `+@groupname` to specify all hosts or all users in the group.

**SYS\$LOGIN:[.SSH2]AUTHORIZATION**

This file contains information on how the server verifies the identity of a user.

**SYS\$LOGIN: [ .SSH2.KNOWNHOSTS ] xxxxyyyy.pub**

These are the public host keys of hosts that a user wants to log in from using "hostbased" authentication (equivalent to the SSH1's RhostsRSAAuthentication). Also, a user must set up his/her individual .SHOSTS or .RHOSTS file. If the username is the same in both hosts, it is adequate to put the public host key in SSH2\_DIR:KNOWNHOSTS and add the host's name to the system-wide SHOSTS.EQUIV or RHOSTS.EQUIV file.

xxxx is the hostname (FQDN) and yyyy denotes the public key algorithm of the key (ssh-dss or ssh-rsa).

For example, bos.example.com's host key algorithm is ssh-dss. The hostkey would then be bos\_example\_com\_ssh-dss.pub in the [ .SSH2.KNOWNHOSTS ] directory.

## SSH2 AUTHORIZATION File Format

The Authorization file contains information on how the server verifies the identity of a user. This file has the same general syntax as the SSH2 configuration files. The following keywords may be used:

Keyword	Description
KEY	The filename of a public key in the [ .SSH2 ] directory in the user's SYS\$LOGIN directory. This key is used for identification when contacting the host. If there are multiple KEY lines, all are acceptable for login.
COMMAND	This keyword, if used, must follow the KEY keyword above. This is used to specify a "forced command" that executes on the server side instead of anything else when the user is authenticated. This option might be useful for restricting certain public keys to perform certain operations.

## SSH2 Logicals

These logicals are used with the SSH server in the system logical name table.

**SSH\_DIR**

Points to the directory where the master server log file is kept. Normally, this is `MULTINET_SPECIFIC_ROOT: [MULTINET]`. It is defined in `START_SSH.COM`.

**SSH\_EXE**

Points to the directory where SSH executables are kept. Normally, this is `MULTINET_COMMON_ROOT: [MULTINET]`. It is defined in `START_SSH.COM`.

**SSH\_LOG**

Points to the directory where the log files are kept. Normally, this is `MULTINET_SPECIFIC_ROOT: [MULTINET.SSH]`. It is defined in `START_SSH.COM`.

**MULTINET\_LOG\_MBX**

Points to the OpenVMS mailbox used to log connection accept and reject messages. This must not be modified by the user.

**MULTINET\_SSH\_ACC\_REJ\_LOG\_FILE**

If the user has set a log file to log connection accept and reject messages, this logical will be defined and will provide the name of the log file. This logical is set by using the `SET LOG-FILE` keyword in `MULTINET CONFIGURE/SERVER`, and should not be modified directly by the user.

**MULTINET\_SSH\_LOG\_ACCEPTS**

When set, causes the server to log successful connection requests as either an OPCOM message or a line in a log file. Specified by the `SET LOG-ACCEPT` command in `MULTINET CONFIGURE/SERVER`. Note that the server does not use the information set in the `ACCEPT-HOSTS` keyword in `CONFIGURE/SERVER`. Rather, it uses the `AllowHosts` and `DenyHosts` keywords in the SSH server configuration file. Also, a successful connection request doesn't equate to a successful authentication request. This logical should not be modified directly by the user.

**MULTINET\_SSH\_LOG\_REJECTS**

When set, causes the server to log rejected connection requests as either an OPCOM message or a line in a log file. Specified by the `SET LOG-REJECT` command in `MULTINET CONFIGURE/SERVER`.

Note that the server does not use the information set in the REJECT-HOSTS keyword in CONFIGURE/SERVER. Rather, it uses the AllowHosts and DenyHosts keywords in the SSH server configuration file. This logical should not be modified directly by the user.

#### **MULTINET\_SSH\_MAX\_SESSIONS**

Set this to the maximum number of concurrent SSH sessions you want to allow on the server system. If MULTINET\_SSH\_MAX\_SESSIONS is not defined, the default is 1000. Setting MULTINET\_SSH\_MAX\_SESSIONS to zero (0) will cause an error. The value must be between 1 and 1000. The suggested place to set this is in START\_SSH.COM. SSH must be restarted to use the new value if it is changed.

#### **SSH\_TERM\_MBX**

Mailbox used by SSHD\_MASTER to receive termination messages from SSHD daemon processes. Do not change this logical name. This is created by the SSHD\_MASTER process.

#### **MULTINET\_SSH\_KEYGEN\_MIN\_PW\_LEN**

Defines the minimum passphrase length when one is to be set in SSHKEYGEN. If not defined, defaults to zero.

#### **MULTINET\_SSH\_PARAMETERS\_n**

These values are set by MultiNet and must not be modified by the user.

#### **MULTINET\_SSH\_USE\_SYSGEN\_LGI**

If defined, causes SSHD to use the VMS SYSGEN value of LGI\_PWD\_TMO to set the login grace time, overriding anything specified in the command line or the configuration file.

#### **MULTINET\_SSH\_ENABLE\_SSH2\_CONNECTIONS**

Enables SSHD Master to accept SSH V2 sessions.

#### **MULTINET\_SSH2\_HOSTKEY\_DIR**

Directory containing the host keys for the SSH V2 server. Normally set to MULTINET\_SPECIFIC\_ROOT:[MULTINET.SSH2.HOSTKEYS]

**MULTINET\_SSH2\_KNOWNHOSTS\_DIR**

Directory containing the public keys for known systems. Normally set to  
MULTINET\_SPECIFIC\_ROOT:[MULTINET.SSH2.KNOWNHOSTS].

**SSH2\_DIR**

Contains all SSH V2-specific files, such as configuration files. Normally set to  
MULTINET\_SPECIFIC\_ROOT:[MULTINET.SSH2]

**SSH Daemon Files**

These files are used by or created by SSH when you log into a daemon. These files are not to be altered in any way.

**SSH\_LOG:SSHD.LOG**

This log file is created by each SSHD daemon.

**SSHD\_MASTER.LOG**

This log file is created by SSHD\_MASTER.

**SSH\_START.COM**

This files is used to start SSH.

# 29. Configuring IPSEC and SETKEY

This chapter describes how to configure the IP Security (IPSEC) and SETKEY protocols. IPSEC provides per-packet authenticity/confidentiality guarantees between peers that communicate using IPSEC.

IPSEC provides security for transmission of sensitive information over unprotected networks such as the Internet. IPSEC acts at the network layer, protecting and authenticating IP packets between participating IPSEC devices.

## About the IP Security (IPSEC) Protocol

IPSEC consists of a couple of separate protocols that are listed below:

**Authentication Header (AH):** Provides authenticity guarantee for packets, by attaching strong crypto checksums to packets. Unlike other protocols, AH covers the whole packet, from the IP header to the end of the packet.

If you receive a packet with AH and the checksum operation was successful, you can be sure about two things *if you and the peer share a secret key, and no other party knows the key*:

- The packet was originated by the expected peer. The packet was not generated by impersonator.
- The packet was not modified in transit.

**Encapsulating Security Payload (ESP):** Provides confidentiality guarantee for packets, by encrypting packets with encryption algorithms. If you receive a packet with ESP and have successfully decrypted it, you can be sure that the packet was not wiretapped in the middle, *provided that you and the peer share a secret key, and no other party knows the key*.

**IP payload compression (IPcomp):** ESP provides encryption service to the packets. However, encryption tends to give negative impact to compression on the wire (such as ppp compression). IPcomp provides a way to compress packets before encryption by ESP. (Of course, you can use IPcomp alone if you wish to).

**Internet Key Exchange (IKE):** An alternate form of keying is called Internet Key Exchange.

**Note:** Security of IPSEC protocols depends on the secrecy of secret keys. If secret keys are compromised, IPSEC protocols can no longer be secure. Take precautions about permission modes of configuration files, key database files, or whatever may lead to information leakage.

# Security Associations and Security Policies

Both ESP and AH rely on security associations. A *security association* (SA) consists of a source, destination and an instruction. The collection of all SA's maintained by the network kernel for a system is termed the *security association database* (SAD). A sample authentication SA may look like this:

```
add 10.0.0.11 10.0.0.216 ah 15700 -A hmac-md5 "1234567890123456";
```

This says “traffic going from 10.0.0.11 to 10.0.0.216 that needs an AH can be signed using HMAC-MD5 using secret 1234567890123456”. This instruction is labelled with Security Parameter Index (SPI) ID 15700 (SPIs are discussed later in this chapter). SAs are symmetrical; both sides of a conversation share exactly the same SA, it is not mirrored on the other side. Note that there is no ‘autoreverse’ rule - this SA only describes a possible authentication from 10.0.0.11 to 10.0.0.216. For two-way traffic, two SAs are needed.

Following is a sample SA:

```
add 10.0.0.11 10.0.0.216 esp 15701 -E 3des-cbc "123456789012123456789012";
```

This says “traffic going from 10.0.0.11 to 10.0.0.216 that needs encryption can be enciphered using 3des-cbc with key 123456789012123456789012”. The SPI ID is 15701.

SAs describe possible instructions, but do not in fact describe policy as to when these need to be used. In fact, there could be an arbitrary number of nearly identical SAs with only differing SPI IDs. To do actual cryptography, we need to describe a *security policy* (SP). This policy can include things such as “use IPSEC if available” or “drop traffic unless we have IPSEC”. The collection of SPs for a system is termed the *security policy database* (SPD).

A typical simple Security Policy (SP) looks like this:

```
spdadd 10.0.0.216 10.0.0.11 any -P out ipsec esp/transport//require  
ah/transport//require;
```



If entered on host 10.0.0.216, this means that all traffic going out to 10.0.0.11 must be encrypted and be wrapped in an AH authenticating header. Note that this does not describe which SA is to be used, that is left as an exercise for the kernel to determine.

In other words, a Security Policy specifies *what* we want; a Security Association describes *how* we want it.

Outgoing packets are labelled with the SA SPI (“the how”) which the kernel used for encryption and authentication so the remote can look up the corresponding verification and decryption instruction.

## IPSEC Configuration File

The configuration file for IPSEC is loaded by the SETKEY program. The default name and location of this file is `MULTINET:IPSEC.CONF`, but other filenames may be used when specifying the `-f` switch for SETKEY.

The configuration file describes all the Security Associations (SA) and Security Policies (SP) for the system. Lines starting with the `#` character are treated as comment lines. Configuration descriptions can be spread over multiple lines in the file. Each description *must* end with a semicolon.

## Configuration File Options

`add [-4n] src dst protocol spi [extensions] algorithm ...;`

Add a SAD entry. This can fail with multiple reasons, including when the key length does not match the specified algorithm.

`get [-4n] src dst protocol spi;`

Show a SAD entry.

`delete [-4n] src dst protocol;`

Remove all SAD entries that match the specification.

`deleteall [-4n] src dst protocol;`

Removes all SAD entries that match the specification.

**flush** [*protocol*];

Clears all SAD entries matched by the options. -F on the command line achieves the same functionality.

**dump** [*protocol*];

Dumps all SAD entries matched by the options. -D on the command line achieves the same functionality.

**spdadd** [-4n] *src\_range dst\_range upperspec policy*;

Adds an SPD entry.

**spdflush** ;

Clears all SPD entries. -FP on the command line achieves the same functionality.

**spddump** ;

Dumps all SPD entries. -DP on the command line achieves the same functionality.

## Configuration File Operation Arguments

Arguments for the configuration file operations are as follows:

### *src/dst*

Source/destination of the secure communication is specified as IPv4/IPv6 addresses. SETKEY can resolve a FQDN into numeric addresses. If the FQDN resolves into multiple addresses, SETKEY will install multiple SAD/SPD entries into the kernel by trying all possible combinations. -4 and -n restricts the address resolution of FQDN in certain ways. -4 restricts results into IPv4/IPv6 addresses only. -n avoids FQDN resolution and requires addresses to be numeric addresses.

### **Protocol**

One of the following:

- `esp`        ESP based on RFC 2406
- `esp-old`    ESP based on RFC 1827
- `ah`         AH based on RFC 2402

- `ah-old` AH based on RFC 1826
- `ipcomp` IPComp

### ***spi***

Security Parameter Index (SPI) for the SAD and the SPD. *spi* must be a decimal number or a hexadecimal number prefixed by '0x'.

SPI values between 0 and 255 are reserved for future use by IANA.

### ***extensions***

May take the following values:

#### ***-m mode***

Specify a security protocol mode for use. Mode is one of the following: `transport`, `tunnel` or `any`. The default value is `any`.

#### ***-r size***

Specify window size in bytes for replay prevention. *size* must be a decimal number in 32-bit word. If *size* is zero or not specified, replay checks don't take place.

#### ***-u id***

Specify the identifier of the policy entry in SPD. See `policy`.

#### ***-f pad\_option***

Defines the content of the ESP padding, and must be one of the following:

- `zero-pad` All of the padding is zero.
- `random-pad` A series of randomized values are set.
- `seq-pad` A series of sequential increasing numbers started from 1 are set.

#### ***-f nocyclic-seq***

Don't allow cyclic sequence number.

#### ***-ls time\_in\_seconds***

Specify the soft lifetime duration of the SA.

#### ***-lh time\_in\_seconds***

Specify the hard lifetime duration of the SA.

### ***Algorithm***

**-E *ealgo key***

Specify an encryption algorithm for ESP.

**-E *ealgo key* -A *aalgo key***

Specify an encryption algorithm *ealgo*, as well as a payload authentication algorithm *aalgo*, for ESP.

**-A *aalgo key***

Specify an authentication algorithm for AH.

**-C *calgo* [-R]**

Specify a compression algorithm for IPComp. If *-R* is specified, the *spi* field value will be used as the IPComp CPI (compression parameter index). If *-R* is not specified, the kernel will use well-known CPI, and the *spi* field will be used only as an index for kernel internal usage.

*key* must be a double-quoted character string, or a series of hexadecimal digits preceded by “0x”.

***src\_range***

***dst\_range***

These are selections of the secure communication specified as an IPv4/IPv6 address or an IPv4/IPv6 address range, and it may also include a TCP/UDP port specification. The addresses may take one of the following forms (an example of each is shown):

<i>Address</i>	192.168.0.15
<i>Address/prefixlen</i>	192.168.0.15/24
<i>Address[port]</i>	192.168.0.15[1234]
<i>Address/prefixlen[port]</i>	192.168.0.15/24[1234]

Note that *prefixlen* and *port* must be decimal numbers. The square brackets around *port* are necessary. For FQDN resolution, the rules applicable to *src* and *dst* apply here as well.

***upperspec***

Upper-layer protocol to be used. *icmp6*, *ip4*, and *any* can be specified, where *any* indicates “any protocol.” The protocol number may also be used.

## ***policy***

One of the following three formats:

```
-P direction discard protocol/mode/src-dst/level[...]
```

```
-P direction none protocol/mode/src-dst/level[...]
```

```
-P direction ipsec protocol/mode/src-dst/level[...]
```

- *direction* must be out or in.
- discard means the packet matching indexes will be discarded.
- none means that IPsec operations will not take place on the packet.
- ipsec means that IPsec operations will take place on the packet.
- *protocol/mode/src-dst/level* specifies the rule as to how the packet will be processed:
  - *protocol* must be one of ah, esp or ipcomp.
  - *mode* must be transport. Note that only transport is valid for the implementation of MultiNet, as tunneling isn't supported.
  - Both *src* and *dst* can be omitted.
  - *level* must be one of the following:
    - default means the kernel consults the system-wide default against the protocol specified.
    - use means that the kernel uses a SA if one is available.
    - require means SA is required whenever the kernel sends a packet matched with the policy.
    - unique is the same as require; in addition, it allows the policy to bind with the unique outbound SA.

# Configuration Encryption Algorithms

The following table shows the supported algorithms that can be used as *aalgo* in a `-A aalgo` protocol parameter:

Algorithm	Key Length in Bits
hmac-md5	128
hmac-sha1	160

keyed-md5	128
keyed-sha1	160
null	0 to 2048

The following table shows the supported algorithms that can be used as *ealgo* in a `-E ealgo` protocol parameter:

Algorithm	Key Length in Bits
blowfish-cbc	40 to 448
cast128-cbc	40 to 128
des-cbc	64
3des-cbc	192
des-deriv	64
rijndael-cbc	128/192/256

Only deflate may be used as *calgo* in a `-C calgo` protocol parameter.

## Simple Configuration Example

What follows is a very simple configuration for talking from host 10.0.0.216 to 10.0.0.11 using encryption and authentication.

On host 10.0.0.216:

```
add 10.0.0.216 10.0.0.11 ah 24500 -A hmac-md5 "1234567890123456";
add 10.0.0.216 10.0.0.11 esp 24501 -E 3des-cbc "12345678901212345678901222";
spdadd 10.0.0.216 10.0.0.11 any -P out ipsec esp/transport//require
ah/transport//require;
```

On host 10.0.0.11, the same Security Associations, no Security Policy:

```
add 10.0.0.216 10.0.0.11 ah 24500 -A hmac-md5 "1234567890123456";
add 10.0.0.216 10.0.0.11 esp 24501 -E 3des-cbc "12345678901212345678901222";
```

With the above configuration in place, \$MU PING 10.0.0.11 from 10.0.0.216 looks like this using TCPDUMP:

```
22:37:52
10.0.0.216>10.0.0.11:AH(spi=0x00005fb4,seq=0xa):ESP(spi=0x00005fb5,seq=0xa)(
DF)
23:37:52 10.0.0.11>10.0.0.216:icmp:echo reply
```

Note how the ping back from 10.0.0.11 is plainly visible. The forward ping cannot be read by TCPDUMP (as the packet is encrypted), but it does show the Security Parameter Index of AH and ESP, which tells 10.0.0.11 how to verify the authenticity of our packet and how to decrypt it.

A problem with the previous example is that it specifies a policy on how 10.0.0.216 should treat packets going to 10.0.0.11, and that it specifies how 10.0.0.11 should treat those packets. However it *does not* instruct 10.0.0.11 to discard unauthenticated or unencrypted traffic. Hence, anybody could insert spoofed and completely unencrypted data and 10.0.0.11 will accept it. To remedy the above, we need an incoming Security Policy on 10.0.0.11, as follows:

```
spdadd 10.0.0.216 10.0.0.11 any -P in ipsec esp/transport//require
ah/transport//require;
```

This instructs 10.0.0.11 that any traffic coming in to it from 10.0.0.216 is required to have valid ESP and AH.

To complete this configuration, the return traffic needs to be encrypted and authenticated as well. Therefore, the following configurations will be required:

On 10.0.0.216:

```
flush;
spdf flush;
# AH
add 10.0.0.11 10.0.0.216 ah 15700 -A hmac-md5 "1234567890123456";
add 10.0.0.11 10.0.0.11 ah 24500 -A hmac-md5 "1234567890123456";
# ESP
add 10.0.0.11 10.0.0.216 esp 15701 -E 3des-cbc "1234567890123456789012";
add 10.0.0.11 10.0.0.216 esp 24501 -E 3des-cbc "1234567890123456789012";
spdadd 10.0.0.216 10.0.0.11 any -P out ipsec esp/transport//require
ah/transport//require;
spdadd 10.0.0.11 10.0.0.216 any -P in ipsec esp/transport//require
ah/transport//require;
```

And on 10.0.0.11:

```
flush;
spdf flush;
# AH
add 10.0.0.11 10.0.0.216 ah 15700 -A hmac-md5 "1234567890123456";
add 10.0.0.11 10.0.0.11 ah 24500 -A hmac-md5 "1234567890123456";
```

```
# ESP
add 10.0.0.11 10.0.0.216 esp 15701 -E 3des-cbc "1234567890123456789012";
add 10.0.0.11 10.0.0.216 esp 24501 -E 3des-cbc "1234567890123456789012";
spdadd 10.0.0.216 10.0.0.11 any -P out ipsec esp/transport//require
ah/transport//require;
spdadd 10.0.0.11 10.0.0.216 any -P in ipsec esp/transport//require
ah/transport//require;
```

Note that in this example, identical keys were used for both directions of traffic. However, it's not required to use identical keys for both directions.

## The SETKEY Program

The configuration file for IPSEC is loaded by the SETKEY program. A foreign command must be defined to invoke SETKEY:

```
$ setkey := $multinet:setkey
```

```
setkey [-v] -f filename
setkey ["-aPlv"] "-D"
setkey ["-Pv"] "-F"
setkey [-h] -x
```

The possible command options for SETKEY are:

-D	Dump the SAD entries. If with -P, the SPD entries are dumped. If with -a, the dead SAD entries will be displayed as well.
-f filename	Read the configuration commands from the specified file.
-F	Flush the SAD entries. If with -P, the SPD entries are flushed.
-a	A dead SAD entry means that it has been expired but remains in the system because it is referenced by some SPD entries.
-h	Add hexadecimal dump on -x mode.
-l	Loop forever with short output on -D.
-v	Be verbose. The program will dump messages exchanged on PF_KEY socket, including messages sent from other processes to the kernel.



-x	Loop forever and dump all the messages transmitted to PF_KEY socket.
----	--

## SETKEY Usage Examples

**Example 1:** Parse and load the policies in the file `MULTINET:IPSEC.CONF` into the kernel (note that the output from parsing can be quite verbose, so part of the output has been deleted from the middle this example to keep it to a reasonable size):

```
$ setkey "-f" multinet:ipsec.conf
Starting parse
Entering state 0
Reducing via rule 1 (line 126), -> commands
state stack now 0
Entering state 1
Reading a token: Next token is 261 (ADD)
Shifting token 261 (ADD), Entering state 2
Reducing via rule 57 (line 537), -> ipaddropts
state stack now 0 1 2
entering state 23
Reading a token: Next token is 292 (STRING)
Shifting token 292 (STRING), Entering state 36
Reducing via rule 61 (line 568), STRING -> ipaddr
state stack now 0 1 2 23
Entering state 39

...
Entering state 19
Reducing via rule 9 (line 141),spdadd_command -> command
state stack now 0 1
Entering state 12
Reducing via rule 2 (line 127), commands command -> commands
state stack now 0
Entering state 1
Reading a token: Now at end of input.
Shifting token 0 ($), Entering state 137
Now at end of input.
```

**Example 2:** Dump out the policies in the kernel:

```
$ setkey "-PD"
192.168.154.10/24[any] 192.168.228.100/24[any]any
  out ipsec
  esp/transport//use
  ah/transport//use
```

```
spid=1 seq=0 pid=149
refcnt=1
```

### Example 3: Dump out the SAD entries in the kernel:

```
$ setkey "-D"
192.168.228.100 192.168.154.10
  ah mode=any spi=1000(0x00002711)reqid=0(0x00000000)
  A:hmac -sha1 6d6f6761 6d6f6761 6d6f6761 6d6f6761
  replay=0 flags=0x00000040 state=mature seq=3 pid=149
  created: Dec 22 15:52:49 2003 current: Dec 22 15:54:30 2003
  diff: 101(s) hard:0(s) soft:0(s)
  last:
    hard:0(s) soft:0(s)
  current:0(bytes) hard:0(bytes) soft:0(bytes)
  allocated:0 hard:0 soft:0
  refcnt=1
192.168.154.10.192 168.228.100
ah mode=any spi=9877(0x00002695)reqid=0(0x00000000)
  A:hmac -sha1 686f6765 686f6765 686f6765 686f6765
  replay=0 flags=0x00000040 state=mature seq=2 pid=149
  created: Dec 22 15:52:49 2003 current: Dec 22 15:54:30 2003
  diff: 101(s) hard:0(s) soft:0(s)
  last:
    hard:0(s) soft:0(s)
  current:0(bytes) hard:0(bytes) soft:0(bytes)
  allocated:0 hard:0 soft:0
  refcnt=1
192.168.228.100.192 168.154.10
ah mode=transport spi=10000(0x00002710)reqid=0(0x00000000)
  E:3des-cbc deadbeef deadbeef deadbeef deadbeef deadbeef
  replay=0 flags=0x00000040 state=mature seq=1 pid=149
  created: Dec 22 15:52:49 2003 current: Dec 22 15:54:30 2003
  diff: 101(s) hard:0(s) soft:0(s)
  last:
    hard:0(s) soft:0(s)
  current:0(bytes) hard:0(bytes) soft:0(bytes)
  allocated:0 hard:0 soft:0
  refcnt=1
192.168.154.10 192.168.228.100
ah mode=transport spi=9876(0x00002694)reqid=0(0x00000000)
  E:3des-cbc 686f6765 686f6765 686f6765 686f6765 686f6765
  replay=0 flags=0x00000040 state=mature seq=0 pid=149
  created: Dec 22 15:52:49 2003 current: Dec 22 15:54:30 2003
  diff: 101(s) hard:0(s) soft:0(s)
  last:
    hard:0(s) soft:0(s)
  current:0(bytes) hard:0(bytes) soft:0(bytes)
  allocated:0 hard:0 soft:0
  refcnt=1
```

### Example 4: Dump the messages out on the PF\_KEY socket.

```
$ setkey "-hx"
14:38:47.009961
00000000:02 0b 00 00 06 00 00 00 00 00 00 00 00 00 00 00
0000000010:02 0b 00 01 02 00 00 00 00 00 00 95 00 00 00
sadb_msg { version=2 type=1 1 errno=0 satype=1
  len=2 reserved=0 seq=0 pid=149
14:38:47 057809
00000000:02 0b 00 01 02 00 00 00 00 00 00 95 00 00 00
```

**Example 5:** Flush all of the entries from the kernel.

```
$ setkey "-F"
$ setkey "-D"
No SAD entries.
```

# IPSEC Configuration File Examples

## Configuration Example: Host-to-Host Encryption

If you want to run host-to-host (transport mode) encryption with manually configured secret keys, the following configuration should be sufficient:

```
# multinet:ipsec.conf
#
# packet will look like this: IPv4 ESP payload
# the node is on 10.1.1.1, peer is on 20.1.1.1
add 10.1.1.1 10.2.1.1 esp 9876 -E 3des-cbc "hogehogehogehogehogehoge";
add 10.2.1.1 10.1.1.1 esp 10000 -E 3des-cbc
0xdeadbeefdeadbeefdeadbeefdeadbeefdeadbeefdeadbeef;
spdadd 10.1.1.1 10.2.1.1 any -P out ipsec esp/transport//use;
```

From 10.1.1.1 to 10.2.1.1, use the 3DES-CBC algorithm with SPI 9876, with secret key "hogehogehogehogehogehoge". The traffic will be identified by SPI 9876.

From 10.2.1.1 to 10.1.1.1, use 3DES-CBC algorithm with SPI 10000, with secret key 0xdeadbeefdeadbeefdeadbeefdeadbeef.

The last line configures per-packet IPSEC policy for the node. Using this configuration, the transmit node (10.1.1.1) used to send packets to the peer (20.1.1.1), is encrypted whenever a secret key is configured in to the kernel. The configuration does not prohibit unencrypted packets from 20.1.1.1 to reach 10.1.1.1. To reject unencrypted packets, the following line would be added to the configuration file:

```
spdadd 10.2.1.1 10.1.1.1 any -P in ipsec esp/transport//require;
```

On the peer's node (10.2.1.1), the configuration will look similar to what is shown in the following example. Note that the addresses need to be swapped on the `spdadd` line, but add lines do not need to be swapped.

```
# multinet:ipsec.conf
#
# packet will look like this: IPv4 ESP payload
# the node is on 10.2.1.1, peer is on 10.1.1.1
add 10.1.1.1 10.2.1.1 esp 9876 -E 3des-cbc "hogehogehogehogehogehoge";
add 10.2.1.1 10.1.1.1 esp 10000 -E 3des-cbc
0xdeadbeefdeadbeefdeadbeefdeadbeefdeadbeefdeadbeefdeadbeef
spdadd 10.2.1.1 10.1.1.1 any -P out ipsec esp/transport//use;
```

The previous example uses human-readable secret keys. However, using human-readable secret keys is discouraged by the IPSEC policy specification, since they are more likely to be compromised than binary keys. Binary keys should be used for normal operations.

Key length is determined by algorithms. See the sections *Authentication Algorithms* and *Encryption Algorithms* for the required key lengths. For 3des-cbc, the secret key *must* be 192 bits (24 bytes). If a shorter or longer key is specified, SETKEY will return an error when parsing the line.

The following is an example of rijndael-cbc (also known as AES) using 128-bit keys.

```
# multinet:ipsec.conf
#
# the packet will look like this: IPv4 ESP payload
# the node is on 10.1.1.1, peer is on 10.2.1.1
# rijndael -cbc with 128bit key
add 10.1.1.1 10.2.1.1 esp 9876 -E rijndael -cbc "hogehogehogehogehoge";
add 10.2.1.1 10.1.1.1 esp 10000 -E rijndael -cbc
0xdeadbeefdeadbeefdeadbeefdeadbeefdeadbeefdeadbeefdeadbeef;
spdadd 10.1.1.1 10.2.1.1 any -P out ipsec esp/transport//use;
```

## Configuration Example: Host-to-Host Authentication

The following example shows a sample configuration for host-to-host authentication:

```
# multinet:ipsec.conf
#
# packet will look like this: IPv4 AH payload
# the node is on 10.1.1.1, peer is on 10.2.1.1
add 10.1.1.1 10.2.1.1 ah 9877 -A hmac-md5 "hogehogehogehogehoge";
add 10.2.1.1 10.1.1.1 ah 10001 -A hmac-md5 "mogamogamogamoga";
spdadd 10.1.1.1 10.2.1.1 any -P out ipsec ah/transport//use;
```

# Configuration Example: Host-to-Host Encryption+Authentication

The following example shows sample keys that are configured for both AH and ESP.

**Note:** It is recommended that you apply AH *after* ESP.

```
# multinet:ipsec.conf
#
# packet will look like this: IPv4 AH ESP payload
# the node is on 10.1.1.1, peer is on 10.2.1.1
add 10.1.1.1 10.2.1.1 esp 9876 -E 3des-cbc "hogehogehogehogehogehoge";
add 10.2.1.1 10.1.1.1 esp 10000 -E 3des-cbc
0xdeadbeefdeadbeefdeadbeefdeadbeefdeadbeefdeadbeef;
add 10.1.1.1 10.2.1.1 ah 9877 -A hmac-md5 "mogamogamogamoga";
spdadd 10.1.1.1 10.2.1.1 any -P out ipsec esp/transport//use
ah/transport//use;
```

## Conformance to Standards and Interoperability

The MultiNet IPSEC implementation conforms to the following IPSEC standards: RFC 2401, RFC 2402, RFC 2404, RFC 2406, RFC 4835, RFC 4868. The IKEv2 (Racoon2) implementation provides support for RFC 4306, RFC 4307 and RFC 4718.

## Racoon Internet Key Exchange Daemon

The RACOON service performs the task of securely creating security associations for participating systems. When a security policy senses the need of a security association, RACOON is notified and securely communicates with an Internet Key Exchange daemon on the other system to establish the security association. Security Policies must still be configured manually with the SETKEY program.

The RACOON service can be controlled through the following MULTINET NETCONTROL commands:

- `DEBUG` - set the debug level for a currently running Racoon IKE daemon.
- `ESTABLISH remote-ip-address [local-ip-address]` – initiate key exchange protocol communication between the remote ip-address and the local ip-address. If local-ip-address is not specified then the value of MULTINET\_HOST\_NAME is used. This does not install Security Associations, but does the initial negotiation necessary to allow Security Associations to be established when necessary. It is not necessary to manually establish the negotiation information – RACOON will do it automatically when necessary.
- `FLUSH` – flush all existing keys
- `NOOP` - No Operation
- `SHOW` – show existing key associations
- `SHUTDOWN` – Shut down the Racoon IKE Daemon. All established keys are flushed as part of this process.
- `START` – Start the Racoon IKE Daemon
- `STOP` – Same as SHUTDOWN
- `VERSION` – Display control interface version

The configuration file (`MULTINET:RACOON.CONF`) can be configured to handle all systems in general, or specific systems.

RACOON negotiates security associations for itself (ISAKMP SA, or phase 1 SA) and for kernel IPsec (IPsec SA, or phase 2 SA). The file consists of a sequence of directives and statements. Each directive is composed by a tag, and statements are enclosed by ‘{’ and ‘}’. Lines beginning with ‘#’ are comments.

## Meta Syntax

Note that you have to pay attention when this section of the manual is describing port numbers. The port number is always enclosed by ‘[’ and ‘]’ (square brackets). In this case, the port number is not an optional keyword. If it is possible to omit the port number, the expression becomes `[[port]]`. Major parameters are listed below.

- *Number* - means a hexadecimal or a decimal number. The former must be prefixed with ‘0x’.
- *string path file* - means any string enclosed in ‘”’ (double quote).
- *Address* - means IPv4 address.
- *Port* - means a TCP/UDP port number. The port number is always enclosed by ‘[’ and ‘]’.
- *Timeunit* - is one of following: `sec`, `secs`, `second`, `seconds`, `min`, `mins`, `minute`, `minutes`, `hour`, `hours`.

## Path Specification

**path include *path*;**

Specifies a path to include a file. See *File Inclusion*.

**path pre\_shared\_key *file*;**

Specifies a file containing pre-shared key(s) for various ID(s). See *Pre-shared Key File*.

**path certificate *path*;**

Racoon will search this directory if a certificate or certificate request is received.

**path backupsa *file*;**

Specifies a file to be stored a SA information which is negotiated by racoon. racoon will install SA(s) from the file with a boot option -B. The file is increasing because racoon simply adds a SA to the file at the moment. You should maintain the file manually.

## File Inclusion

**include *file***

Other configuration files can be included.

## Timer Specification

**timer { *statements* }**

Specifies various timer values.

**counter *number*;**

The maximum number of retries to send. The default is 5.

**interval *number timeunit*;**

The interval to resend, in seconds. The default time is 10 seconds.

**persend *number*;**

The number of packets per send. The default is 1.

**phase1 *number timeunit*;**

The maximum time it should take to complete phase 1. The default time is 15 seconds.

**phase2 number timeuni;**

The maximum time it should take to complete phase 2. The default time is 10 seconds.

## Listening Port Specification

**listen { statements }**

If no listen directive is specified, racoon will listen on all of the available interface addresses. The following is the list of valid statements:

**isakmp address [[port]];**

If this is specified, racoon will only listen on *address*. The default port is 500, which is specified by IANA. You can provide more than one address definition.

**strict\_address;**

Require that all addresses for ISAKMP must be bound. This statement will be ignored if you do not specify any addresses.

## Remote Nodes Specifications

**remote (address | anonymous) [[port]] { statements }**

Specifies the parameters for IKE phase 1 for each remote node. The default port is 500. If **anonymous** is specified, the statements apply to all peers which do not match any other remote directive.

The following are valid statements:

**exchange\_mode (main | aggressive | base);**

Defines the exchange mode for phase 1 when racoon is the initiator. It also defines the acceptable exchange mode when racoon is the responder. More than one mode can be specified by separating them with a comma. All of the modes are acceptable. The first exchange mode is what racoon uses when it is the initiator.

**doi ipsec\_doi;**

Use IPSEC-DOI as specified RFC 2407. You can omit this statement.

**situation identity\_only;**

Use SIT\_IDENTITY\_ONLY as specified RFC 2407. You can omit this statement.



**my\_identifier idtype ...;**

Specifies the identifier sent to the remote host and the type to use in the phase 1 negotiation. `address`, `fqdn`, `user_fqdn`, `keyid` and `asn1dn` can be used as an *idtype*. they are used like:

<code>my_identifier address [address];</code>	The type is the IP address. This is the default type if you do not specify an identifier to use.
<code>my_identifier user_fqdn string;</code>	The type is a USER_FQDN (user fully-qualified domain name).
<code>my_identifier fqdn string;</code>	The type is a FQDN (fully-qualified domain name).
<code>my_identifier keyid file;</code>	The type is a KEY_ID.
<code>my_identifier asn1dn [string];</code>	The type is an ASN.1 distinguished name. If <code>string</code> is omitted, racoon will get DN from Subject field in the certificate.

**peers\_identifier idtype ...;**

Specifies the peer's identifier to be received. If it is not defined then racoon will not verify the peer's identifier in ID payload transmitted from the peer. If it is defined, the behavior of the verification depends on the flag of `verify_identifier`. The usage of *idtype* is same to `my_identifier`.

**verify\_identifier (on | off);**

If you want to verify the peer's identifier, set this to `on`. In this case, if the value defined by `peers_identifier` is not same to the peer's identifier in the ID payload, the negotiation will fail. The default is `off`.

**certificate\_type certspec;**

Specifies a certificate specification. *certspec* is one of following:

<code>x509 certfile privkeyfile;</code>	<i>certfile</i> is the file name of certificate. <i>privkeyfile</i> is the file name of secret key.
<code>peers_certfile (dnssec   certfile);</code>	If <code>dnssec</code> is defined, racoon will ignore the certificate payload from the peer, and try to get the peer's certificate from DNS instead. If <code>certfile</code> is defined, racoon will ignore the CERT payload from

	the peer, and will use this certificate as the peer's certificate.
<code>send_cert (on   off);</code>	If you do not want to send a certificate for some reason, set this to <code>off</code> . The default is <code>on</code> .
<code>send_cr (on   off);</code>	If you do not want to send a certificate request for some reason, set this to <code>off</code> . The default is <code>on</code> .
<code>verify_cert (on   off);</code>	If you do not want to verify the peer's certificate for some reason, set this to <code>off</code> . The default is <code>on</code> .

**`lifetime time number timeunit;`**

Define a lifetime of a certain time which will be proposed in the phase 1 negotiations. Any proposal will be accepted, and the attribute(s) will be not proposed to the peer if you do not specify it(them). They can be individually specified in each proposal.

**`initial_contact (on | off);`**

Enable this to send an INITIAL-CONTACT message. The default value is `on`. This message is useful only when the implementation of the responder chooses an old SA when there are multiple SAs which are different established time, and the initiator reboots. If racoon did not use the message, the responder would use an old SA even when a new SA was established. The KAME stack has the switch in the system wide value, `net.key.preferred_oldsas`. When the value is zero, the stack always uses a new SA.

**`passive (on | off);`**

If you do not want to initiate the negotiation, set this to `on`. The default value is `off`. It is useful for a server.

**`proposal_check level;`**

Specifies the action of lifetime length and PFS of the phase 2 selection on the responder side. The default level is `strict`. If the level is:

- `Obey` The responder will obey the initiator anytime.
- `Strict` If the responder's length is longer than the initiator's one, the responder uses the initiator's one. Otherwise it rejects the proposal. If PFS is not required by the responder, the responder will obey the proposal. If PFS is required by both sides and if the responder's group is not equal to the initiator's one, then the responder will reject the proposal.

- **Claim** If the responder's length is longer than the initiator's one, the responder will use the initiator's one. If the responder's length is shorter than the initiator's one, the responder uses its own length *and* sends a RESPONDER-LIFETIME notify message to an initiator in the case of lifetime. About PFS, this directive is same as `strict`.
- **Exact** If the initiator's length is not equal to the responder's one, the responder will reject the proposal. If PFS is required by both sides and if the responder's group is not equal to the initiator's one, then the responder will reject the proposal.

**support\_mip6 (on | off);**

If this value is set on then both values of ID payloads in phase 2 exchange are always used as the addresses of end-point of IPsec-SAs. The default is `off`.

**generate\_policy (on | off);**

This directive is for the responder. Therefore, you should set `passive on` in order that `racoon` only becomes a responder. If the responder does not have any policy in SPD during phase 2 negotiation, and the directive is set on, then `racoon` will choice the first proposal in the SA payload from the initiator, and generate policy entries from the proposal. It is useful to negotiate with the client which is allocated IP address dynamically. Note that inappropriate policy might be installed by the initiator because the responder just installs policies as the initiator proposes. So that other communication might fail if such policies installed. This directive is ignored in the initiator case. The default value is `off`.

**nonce\_size number;**

Define the byte size of nonce value. `Racoon` can send any value although RFC2409 specifies that the value MUST be between 8 and 256 bytes. The default size is 16 bytes.

**proposal { sub-substatements }**

**encryption\_algorithm algorithm;**

Specify the encryption algorithm used for the phase 1 negotiation. This directive must be defined. *algorithm* is one of following: `des`, `3des`, `blowfish`, `cast128`. For other transforms, this statement should not be used.

**hash\_algorithm algorithm;**

Define the hash algorithm used for the phase 1 negotiation. This directive must be defined. *algorithm* is one of following: `md5`, `sha1`.

**authentication\_method type;**

Defines the authentication method used for the phase 1 negotiation. This directive must be defined. *type* is one of: *pre\_shared\_key*, *rsasig*, *gssapi\_krb*.

***dh\_group group;***

Define the group used for the Diffie-Hellman exponentiations. This directive must be defined. *group* is one of following: *modp768*, *modp1024*, *modp1536*. Or you can define 1, 2, or 5 as the DH group number. When you want to use aggressive mode, you must define same DH group in each proposal.

***lifetime time number timeunit;***

Define lifetime of the phase 1 SA proposal. Refer to the description of *lifetime* directive immediately defined in remote directive.

***gssapi\_id string;***

Define the GSS-API endpoint name, to be included as an attribute in the SA, if the *gssapi\_krb* authentication method is used. If this is not defined, the default value of *ike/hostname* is used, where *hostname* is the FQDN of the interface being used.

## Policy Specifications

The policy directive is obsolete, policies are now in the SPD. *racoon* will obey the policy configured into the kernel by *setkey*, and will construct phase 2 proposals by combining *sainfo* specifications in *racoon.conf*, and policies in the kernel.

## Sainfo Specifications

***sainfo (source\_id destination\_id | anonymous) { statements }***

Defines the parameters of the IKE phase 2 (IPsec-SA establishment). *source\_id* and *destination\_id* are constructed like:

```
address address [/ prefix] [[port]] ul_proto
```

or

```
idtype string
```

The statements must exactly match the content of ID payload. This is not like a filter rule. For example, if you define *3ffe:501:4819::/48* as *source\_id*. *3ffe:501:4819:1000:/64* will not match.

**pfs\_group group;**

Defines the group of Diffie-Hellman exponentiations. If you do not require PFS then you can omit this directive. Any proposal will be accepted if you do not specify one. group is one of following: modp768, modp1024, modp1536. Or you can define 1, 2, or 5 as the DH group number.

**lifetime time number timeunit;**

Defines the lifetime of amount of time which are to be used IPsec-SA. Any proposal will be accepted, and no attribute(s) will be proposed to the peer if you do not specify it(them). See the proposal\_check directive.

Racoon does not have the list of security protocols to be negotiated. The list of security protocols are passed by SPD to the kernel. Therefore, you have to define all of the potential algorithms in the phase 2 proposals even if there is an algorithm which will not be used. These algorithms are defined by using the following three directives, and they are lined with single comma as the separator. For algorithms that can take variable-length keys, algorithm names can be followed by a key length, like ``blowfish 448". racoon will compute the actual phase 2 proposals by computing the permutation of the specified algorithms, and then combining them with the security protocol specified by the SPD. For example, if des, 3des, hmac\_md5, and hmac\_sha1 are specified as algorithms, we have four combinations for use with ESP, and two for AH. Then, based on the SPD settings, racoon will construct the actual proposals. If the SPD entry asks for ESP only, there will be 4 proposals. If it asks for both AH and ESP, there will be 8 proposals. Note that the kernel may not support the algorithm you have specified.

**encryption\_algorithm algorithms;**

des, 3des, des\_iv64, des\_iv32, rc5, rc4, idea, 3idea, cast128, blowfish, null\_enc, twofish, rijndael (used with ESP)

**authentication\_algorithm algorithms;**

des, 3des, des\_iv64, des\_iv32, hmac\_md5, hmac\_sha1, non\_auth (used with ESP authentication and AH)

**compression\_algorithm algorithms;**

deflate (used with IPComp)

**log level;**

Defines the logging level. level is one of following: notify, debug and debug2. The default is notify. If you put too high logging level on slower machines, IKE negotiation can fail due to timing constraint changes.

### **padding { statements }**

Specifies the padding format. The following are valid statements:

**randomize (on | off);**

Use a randomized value for padding. The default is `on`.

**randomize\_length (on | off);**

Use a random pad length. The default is `off`.

**maximum\_length number;**

Defines a maximum padding length. If `randomize_length` is `off`, this is ignored. The default is 20 bytes.

**exclusive\_tail (on | off);**

Puts the number of pad bytes minus one into last part of the padding. The default is `on`.

**strict\_check (on | off);**

Constrain the peer to set the number of pad bytes. The default is `off`.

**complex\_bundle (on | off);**

Sets the interpretation of proposal in the case of SA bundle. Normally “IP AH ESP IP payload” is proposed as “AH tunnel and ESP tunnel”. The interpretation is more common to other IKE implementations; however, it allows very limited set of combinations for proposals. With the option enabled, it will be proposed as “AH transport and ESP tunnel”. The default value is `off`.

### **Pre-shared key File**

Pre-shared key file defines a pair of the identifier and the shared secret key which are used at Pre-shared key authentication method in phase 1. The pair in each line are separated by some number of blanks and/or tab characters. Key can be included any blanks because all of the words after the 2nd column are interpreted as a secret key. Lines starting with ‘#’ are ignored. Keys which start with ‘0x’ are hexadecimal strings. Note that the file must be owned by the user ID running racoon (usually SYSTEM), and must not be accessible by others.

## **Example RACOON configuration file:**

```
#
# Basic Racoon configuration file
#
path pre_shared_key "multinet:psk.txt" ;
remote anonymous
```

```

{
exchange_mode aggressive, main, base ;
lifetime time 24 hour ;
proposal
{
encryption_algorithm blowfish 448;
hash_algorithm md5; #sha1;
authentication_method pre_shared_key ;
dh_group 5 ;
}
}

sainfo anonymous
{
pfs_group 2;
lifetime time 12 hour ;
encryption_algorithm blowfish 448;
authentication_algorithm hmac_sha1, hmac_md5 ;
compression_algorithm deflate ;
}

```

## Example pre-shared key file:

```

192.168.1.2 deadbeef
192.168.1.3 deadbeef
192.168.1.4 deadbeef
192.168.1.5 face0ff0

```

## Restrictions

The following restrictions exist regarding the use of IPSEC in MultiNet 5.6. These restrictions may be lifted in future releases of MultiNet.

- Security may not be set on a socket-by-socket basis via the use of `setsockopt()`.
- Only transport mode is supported to both AH and ESP. Tunnel mode (primarily used for VPN's) is not supported in any mode (AH or ESP).
- IPcomp is not currently implemented.

# IPSec key management with Racoon2

Racoon2 is the new IPSEC key management package that replaces Racoon. MultiNet Racoon2 is available on OpenVMS Alpha and Integrity platforms and offers IKEv1 in main mode and IKEv2. Racoon2 consists of three images:

- SPMD - the Security Policy Management Daemon, which installs security policies and acts on various requests from IKED
- SPMDCTL - the program to send SPMD control messages while it is running
- IKED - the IPSEC Key Exchange Daemon, which authenticates the identification of a remote system and establishes security associations.

IKED supports the following specifications:

- Internet Key Exchange (IKEv2) Protocol
- RFC 4306, Internet Key Exchange (IKEv2) Protocol
- RFC 4307, Cryptographic Algorithms for Use in the Internet Key Exchange Version 2 (IKEv2)
- RFC 4718, IKEv2 Clarifications and Implementation Guidelines
- The Internet Key Exchange (IKE)
- RFC 2409, The Internet Key Exchange (IKE)
- RFC 3947, Negotiation of NAT-Traversal in the IKE
- RFC 3948, UDP Encapsulation of IPsec ESP Packets

All images assume a default configuration file of `MULTINET:RACOON2.CONF`

Additional information on Racoon2 is available from <http://www.racoon2.wide.ad.jp/>.

**Note:** KINKD is not provided because there are a number of Kerberos routines that it needs that are not visible in the packages provided for OpenVMS.

## SPMD

SPMD manages IPsec Security Policy for racoon2, the key management daemon (IKED). It can also serve as a local DNS proxy server if you set on in racoon2 configuration file.





```

        sp_src_ipaddr sp_dst_ipaddr \
        [sa_src_ipaddr sa_dst_ipaddr]
                : add policy
policy delete selector_index
                : delete policy
policy dump : show policies under spmd management
interactive                : process only login
        shutdown                : tell spmd to shutdown
status                    : show statistics

```

**Note:** When using command line options, make sure to enclose them in double quotes to preserve case.

## IKED

IKED is a key management daemon, which supports the Internet Key Exchange (IKE) protocol version 1 (RFC2409) and version 2 (RFC4306). It is driven by messages from the kernel via the PF\_KEYv2 interface or by negotiation requests from remote peers, and manages IPSec security associations according to `racoon2.conf`. Users who are familiar with IKED on Unix platforms will notice that IKED on MultiNet only runs in “foreground mode”.

## Usage

```

iked [-f file] [-p port] [-46] [-I address] \
    [-S selector_index] [-D number] [-dvVh] [-l logfile]
-f specify the configuration file.
-p specify the isakmp port number to listen to.
-4 use IPv4 only.
-6 use IPv6 only.
-I immediately initiate to the address specified.
-S immediately initiate using the selector specified.
-D specify the debug flags.
-d increase the debug level.
-v specify to output messages to standard error,
  in addition to syslog.
-V show the iked version.
-h show this help.
-l specify log output file (instead of syslog).

```

Debug option:

```

0x0001  DEBUG_FLAG_DEBUG      log debug messages.
0x0002  DEBUG_FLAG_TRACE     show internal processing trace.
0x0004  DEBUG_FLAG_CONFIG    show config parsing trace.

```

0x0008 DEBUG\_FLAG\_PFKY PFKEY and SPMIF are ignored.

**Note:** When using command line options, make sure to enclose them in double quotes to preserve case.

## Authentication with Pre-shared Keys

IKED uses the entire contents of the PSK file as the key, so watch for any trailing newlines that might get inadvertently added. This is different from Racoon, which uses a PSK file that is keyed by IP Address to designate the key.

## Authentication with Certificates

The following steps are required to get two hosts using Racoon2 IKED to authenticate each other with certificates when using the IKEv1 or IKEv2 protocol:

- Create a Certificate Signing Request with an unencrypted private key
- Send the CSR file to the Certificate Authority (CA) for signing. Make sure that the CA includes the following information in the CRT (or PEM) file that is returned:
  - SubjectAltName = DNS:*FQDN*
  - SubjectAltName = email:*email\_address\_of\_contact*
  - SubjectAltName = IP:*ip\_address\_of\_system*

If the VMS OpenSSL software is used, then the above information needs to be added to the `SSL$CONF:SSL$CA.CNF` file before signing each certificate under the `[CA_x509_extensions]` section.

- Convert the CRT files to PEM files (if necessary) with the following commands:
- `$ openssl x509 -in host.crt -out host.der -outform DER`
- `$ openssl x509 -in host.der -inform DER -out host.pem -outform PEM`
- Hash certificates (PEM files) (option 9 with `SSL$COM:SSL$CERT_TOOL`)
- Distribute the PEM files and hashed files to the hosts involved.

The default format for the `SubjectAltName` IP address is a binary value. If your CA encodes this as a text string then define the logical `MULTINET_RACOON2_BINARY_IPV4_SUBJECTALTNAME` `FALSE` (or `MULTINET_RACOON2_BINARY_IPV6_SUBJECTALTNAME` for IPv6).

In addition to the above, IKEv2 needs the following in the Racoon2 configuration:

```
Send_peers_id on;
```

The OpenSSL code that IKED is built with looks in `SSLCERTS:` (not `SSL$CERTS:`) for the CA certificate, so you need to define the logical `SSLCERTS` to point to the directory that contains the CA certificate and any intermediate certificates in the chain.

## Scripts

Command files (or scripts) can be invoked when certain events occur. The command files are passed up to 8 parameters in the following format:

```
Parameter_name=parameter_value
```

Possible parameter names:

- `LOCAL_ADDR`
- `LOCAL_PORT`
- `REMOTE_ADDR`
- `REMOTE_PORT`
- `SELECTOR_INDEX`
- `IPSEC_MODE`
- `UPPER_LAYER_PROTOCOL`
- `INTERNAL_ADDR4`
- `APPLICATION_VERSION`
- `OLD_SRC`
- `OLD_DST`
- `NEW_SRC`
- `NEW_DST`
- `INTERNAL_ADDR`
- `INTERNAL_DNS4`
- `INTERNAL_WINS4`
- `INTERNAL_DHCP4`
- `INTERNAL_ADDR6`
- `INTERNAL_DNS6`
- `INTERNAL_DHCP6`
- `LOCAL_NET_ADDR`
- `LOCAL_NET_PREFIXLEN`
- `LOCAL_NET_PORT`
- `REMOTE_NET_ADDR`
- `REMOTE_NET_PREFIXLEN`

- `REMOTE_NET_PORT`

See *Scripts* under *Directive Details* for more information about which parameters can be used with which scripts.

## Compatibility with Racoon

The MultiNet implementation of `IKED` has been modified so that it works with Racoon with more encryption keys than the typical `IKED` implementation would. Unfortunately, these changes will prevent it from working with other implementations of `IKED` in `IKEv1` mode for some encryption methods. To disable this change, define the following logical to `No/False/0` before starting `IKED`:

```
$ DEFINE/SYSTEM MULTINET_RACOON2_IKEV1_MORE_DEFAULT_KEYLENS NO
```

## Troubleshooting

The first step in troubleshooting is to start `IKED` with `-D7`. If possible, this should be done on both sides of the connection because one side may provide more useful information than the other as to why the security associations could not be negotiated. Potential causes of problems are:

- Identification parameters (FQDN and IPADDR)
- Authentication (pre-shared key or certificate)
- IPsec requirements (requested combinations of ESP and AH, encryption and hash algorithms.)

For `IKEv1` there may be a set of lines similar to:

```
2020-03-24 09:34:24 [DEBUG]: IPSEC_Doi.C;23:372: Compared: DB:Peer
2020-03-24 09:34:24 [DEBUG]: IPSEC_Doi.C;23:373: (lifetime = 28800:600)
2020-03-24 09:34:24 [DEBUG]: IPSEC_Doi.C;23:376: (lifebyte = 0:0)
2020-03-24 09:34:24 [DEBUG]: IPSEC_Doi.C;23:382: enctype = 3DES-CBC:3DES-CBC
2020-03-24 09:34:24 [DEBUG]: IPSEC_Doi.C;23:387: (encklen = 0:0)
2020-03-24 09:34:24 [DEBUG]: IPSEC_Doi.C;23:389: hashtype = MD5:SHA1
2020-03-24 09:34:24 [DEBUG]: IPSEC_Doi.C;23:394: authmethod = pre-shared
key:pre-shared key
2020-03-24 09:34:24 [DEBUG]: IPSEC_Doi.C;23:399: dh_group = 1536-bit MODP
group:1536-bit MODP group
2020-03-24 09:34:24 [DEBUG]: IPSEC_Doi.C;23:271: no suitable proposal found.
```

The above information points out the differences in the `IKEv1` transport configurations between the two systems. The values compared are a combination of program defaults, values in the default configuration file, and values in the `transport_ike` configuration file, so all configuration files must be checked for the differences.

# PSKGEN

PSKGEN is a simple program to write a text string to a specified file such that there is no record information in the file and no extraneous carriage control characters. This makes the file compatible with a file that might be generated on a Unix system. Use PSKGEN to generate a pre-shared key file, if that is the authentication method you have chosen. To use the program, define a symbol:

```
$ pskgen := $multinet:pskgen
$ pskgen multinet:racoons2_psk.txt deadbeef
```

## Usage

```
pskgen file_specification pre_shared_key_text
```

# Starting Racoon2 on MultiNet

1. Copy configuration file templates and edit them:

```
$ copy multinet:racoons2_conf.template multinet:racoons2.conf
etc...
```

2. Start SPMD with the following command lines

```
$ spmd := $multinet:spmd
$ spawn/nwait spmd -f multinet:racoons2.conf
```

3. Start IKED with the following command lines

```
$ iked := $multinet:iked
$ iked -f multinet:racoons2.conf
```

Steps 2 and 3 above may be issued by hand in a terminal window. If you wish this to be started automatically with MultiNet, submit START\_SPMD.COM as a batch job from MULTINET:LOCAL\_INITIALIZATION.COM using the /after qualifier with a delta time such that SPMD/IKED startup takes place after MultiNet startup has completed. A future release of MultiNet will have this integrated into the MultiNet Master Server.

```
$ submit/noprint/after="+0:1:00" multinet:start_spmd.com
```

The above line placed in MULTINET:LOCAL\_INITIALIZATION.COM will submit to SYS\$BATCH the procedure START\_SPMD.COM for starting in 1 minute. You may need to adjust the delta time depending on your system configuration. It may also be necessary to uncomment the wait command between the startup of SPMD and IKED in START\_SPMD.COM.

# Sample Configuration

The sample configuration files below are intended to show some of the functionality available, and do not illustrate the complete configuration language.

## racoon2.conf

```
# Edit racoon2_vals.conf for your environment
include "multinet:racoon2_vals.conf";
# interface info
interface
{
    ike {
        MY_IP port 500;
    };
#
# For VMS specify loopback address and port number.
#
    spmd {
        127.0.0.1 port 5500;
    };
    spmd_password "multinet:spmd.pwd";
};

# resolver info
resolver
{
    resolver off;
#
    resolver on;
#
    nameserver {
#
        192.168.0.3 port 53;
#
    };
#
    dns_query {
#
        127.0.0.1 port 53;
#
        ::1 port 53;
#
    };
};

#
# This line includes default configuration file;
# Please don't touch this line (especially novice user);
#

include "multinet:racoon2_default.conf";

## Transport mode IKEv2 or IKEv1
include "multinet:transport_ike.conf";
```

## racoon2\_vals.conf

```
setval {
### Directory Settings ###
# Preshared key file directory : specify if you want to use preshared
# keys
    PSKDIR            "multinet:";

    # Cert file directory : specify if you want to use certs
    CERTDIR           "SSL$CERTS:";

### ID Settings ###
    # your FQDN : specify if you want to use FQDN as your ID
    MY_FQDN            "client.example.com";
    # Peer's FQDN : specify if you want to use FQDN as peer's ID
    PEERS_FQDN         "server.example.com";
### Preshared Key Setting ###
    # Preshared Key file name
    # You can generate it by pskgen.
    PRESHRD_KEY       "psk2.txt";

### Certificate Setting ###
    # Set following parameters if you use certificates in
    # IKE negotiation and
    # _SET_ 'kmp_auth_method { rsasig;};' in each remote{}
    # section of tunnel_ike{_natt}.conf/transport_ike.conf
    # files.
    # For more information, please see USAGE.
    #
    # Your Public Key file name
    MY_PUB_KEY         "client.pem";

    # Your Private Key file name
    MY_PRI_KEY         "client.key";

    # Peer's Public Key file name
    PEERS_PUB_KEY      "server.pem";

### Transport Mode Settings ###

    # Your IP Address

    MY_IPADDRESS       "192.168.0.1";

    # Peer's IP Address
    PEERS_IPADDRESS    "198.168.0.2";

### Configuration Payload Settings (for IKEv2)###
    # IPv4 Address Pool For Assignment
    CP_ADDRPL4_START   "10.7.73.128";
    CP_ADDRPL4_END     "10.7.73.254";
```



```

# IPv6 Address Pool For Assignment
CP_ADDRPL6_START "fd01::1000";
CP_ADDRPL6_END   "fd02::2000";

# DNS Server Address(es) (ex. "10.7.73.1; 10.7.73.2")
CP_DNS           "10.7.73.1";

# DHCP Server Address(es)
CP_DHCP          "10.7.73.1";

# Application Version String
CP_APPVER        "Racoon2 iked"

### Scripts
## IKEv2
# IKESAUP_SCR     "multinet:ikesa-up.com";
# IKESADOWN_SCR   "multinet:ikesa-down.com";
# CHILDDUP_SCR    "multinet:child-up.com";
# CHILDDOWN_SCR   "multinet:child-down.com";
# IKESAREKEY_SCR  "multinet:ikesa-rekey.com";
# CHILDREKEY_SCR  "multinet:child-rekey.com";
# MIGRATION_SCR   "multinet:migration.com";
## IKEv1
# PH1UP_SCR       "multinet:ph1-up.com";
# PH1DOWN_SCR     "multinet:ph1-down.com";

racoon2_default.conf

#
# default section
#
default
{
    remote {
        acceptable_kmp { ikev2; ikev1; };
        ikev1 {
            logmode normal;
            kmp_sa_lifetime_time 600 sec;
            kmp_sa_lifetime_byte infinite;
            interval_to_send 10 sec;
            times_per_send 1;
            ipsec_sa_nego_time_limit 40 sec;
            kmp_enc_alg { 3des_cbc; };
            kmp_hash_alg { sha1; md5; };
            kmp_dh_group { modp3072; modp2048; modp1024; modp1536;};
            kmp_auth_method { psk; };
            random_pad_content off;
        };
        ikev2 {
            logmode normal;
            kmp_sa_lifetime_time infinite;
            kmp_sa_lifetime_byte infinite;

```

```

        max_retry_to_send 3;
        interval_to_send 10 sec;
        times_per_send 1;
        kmp_sa_nego_time_limit 60 sec;
        ipsec_sa_nego_time_limit 40 sec;
        kmp_enc_alg { 3des_cbc; };
        kmp_prf_alg { hmac_md5; hmac_sha1; };
        kmp_hash_alg { hmac_sha1; hmac_md5; };
        kmp_dh_group { modp3072; modp2048; modp1024; };
        kmp_auth_method { psk; };
        random_pad_content on;
        random_padlen on;
        max_padlen 50 bytes;
    };
};

policy {
    ipsec_mode transport;
    ipsec_level require;
};

ipsec {
    ipsec_sa_lifetime_time infinite;
    ipsec_sa_lifetime_byte infinite;
};

sa {
    esp_enc_alg { 3des_cbc; };
    esp_auth_alg { hmac_sha1; hmac_md5; };
};

ipsec ipsec_ah_esp {
    ipsec_sa_lifetime_time 28800 sec;
    sa_index { ah_01; esp_01; };
};

ipsec ipsec_esp {
    ipsec_sa_lifetime_time 28800 sec;
    sa_index esp_01;
};

sa ah_01 {
    sa_protocol ah;
    ah_auth_alg { hmac_sha1; hmac_md5; };
};

sa esp_01 {
    sa_protocol esp;
    esp_enc_alg { 3des_cbc; };
    esp_auth_alg { hmac_sha1; hmac_md5; };
};

```

```

transport_ike.conf

# ike transport mode (esp/tcp)
remote ike_trans_remote {
    acceptable_kmp { ikev2; ikev1;};
    ikev2 {
#         my_id fqdn "${MY_FQDN}";
#         my_id ipaddr "${MY_IPADDRESS}";
#         peers_id fqdn "${PEERS_FQDN}";
#         peers_id ipaddr "${PEERS_IPADDRESS}";
#         peers_ipaddr "${PEERS_IPADDRESS}" port 500;
#         kmp_sa_nego_time_limit 600 sec;
#         # ipsec_sa_nego_time_limit 360 sec;
#         ## Use Preshared Key
#         kmp_auth_method { psk; };
#         pre_shared_key "${PSKDIR}${PRESHRD_KEY}";
#         ## Use Certificate
#         kmp_auth_method { rsasig; };
#         my_public_key x509pem "${CERTDIR}${MY_PUB_KEY}"
"${CERTDIR}${MY_PRI_KEY}";
#         peers_public_key x509pem "${CERTDIR}${PEERS_PUB_KEY}" "";
#         send_peers_id on;
#         script {
#             phase1_up          "${PH1UP_SCR}";
#             phase1_down        "${PH1DOWN_SCR}";
#         };
    };
    ikev1 {
#         my_id ipaddr "${MY_IPADDRESS}";
#         peers_id ipaddr "${PEERS_IPADDRESS}";
#         kmp_auth_method {psk; };
#         pre_shared_key "${PSKDIR}${PRESHRD_KEY}";
#         kmp_auth_method { rsasig; };
#         my_public_key x509pem "${CERTDIR}${MY_PUB_KEY}"
"${CERTDIR}${MY_PRI_KEY}";
#         peers_public_key x509pem "${CERTDIR}${PEERS_PUB_KEY}" "";
#         peers_ipaddr "${PEERS_IPADDRESS}" port 500;
#         nonce_size 16;
#         kmp_hash_alg {sha1; md5;};
#         initial_contact on;
#         script {
#             phase1_up          "${PH1UP_SCR}";
#             phase1_down        "${PH1DOWN_SCR}";
#         };
    };
    selector_index ike_trans_sel_in;
};

selector ike_trans_sel_out {

```

```
direction outbound;
src "${MY_IPADDRESS}";
dst "${PEERS_IPADDRESS}";
upper_layer_protocol "tcp";
policy_index ike_trans_policy;
};

selector ike_trans_sel_in {
direction inbound;
dst "${MY_IPADDRESS}";
src "${PEERS_IPADDRESS}";
upper_layer_protocol "tcp";
policy_index ike_trans_policy;
};

policy ike_trans_policy {
action auto_ipsec;
remote_index ike_trans_remote;
ipsec_mode transport;
ipsec_index { ipsec_esp; };
ipsec_level require;
};
```

# 30. Intrusion Prevention System (IPS)

This chapter describes the MultiNet Intrusion Prevention System (IPS). This security feature monitors network and/or system activities for malicious or unwanted behavior and can react, in real-time, to block or prevent those activities. IPS is highly flexible and customizable. When an attack is detected, pre-configured rules will block an intruder's IP address from accessing the MultiNet system, prevent an intruder from accessing a specific application, or both. The time period that the filter is in place is configurable. An API is provided so that MultiNet customers can incorporate the IPS functionality into user-written applications.

IPS is implemented by instrumenting components (e.g, MultiNet SSH or FTP, or user-supplied components) with a Process Software-supplied API that allows them to report events, such as invalid login attempts, to the `FILTER_SERVER` process. The filter server, started when MultiNet starts, maintains the component rulesets and lists of events, based on the originating address for the offending connection, and when defined limits are reached, creates and sets timed filters in the MultiNet kernel to filter that traffic.

Note: For a gentle introduction to setting up and configuring IPS, please see [http://www.process.com/products/multinet/whitepapers/using\\_ips.html](http://www.process.com/products/multinet/whitepapers/using_ips.html)

## IPS Operation

All of the operating parameters such as the definition of rule, the number of events/unit time to trigger a filter, the duration of a filter, etc. are all defined by component configuration files.

Events are recorded per source address, per rule, per destination address, per component. This provides the ability to have differing filtering criteria for different interfaces (for example, an internal network vs. an external network). Addresses or networks may be excluded from consideration when an event is logged. This feature allows, for example, different settings to be used for internal vs. external networks.

Events “age”; after a time period, old events are discarded from the list of events so that infrequent event occurrences (e.g., mistyping a password) have less chance of inadvertently causing a filter to be set.

Note that when a filter is triggered for an address and rule, the list of known events for that rule and address are deleted.

Multiple filters may be set in sequence for a component/rule/source address/destination address as events are logged. The purpose of this is to make a filter progressively longer. For example, the first filter set for an address and rule might be 5 minutes long; the next, 10 minutes long; the next, 15 minutes long; etc., up to 5 filter times.

# Configuring IPS

IPS is configured in two steps:

1. Configuring the main process-specific parameters of the `FILTER_SERVER` process (for example, the size of the mailbox used by applications to communicate with the `FILTER_SERVER` process).
2. Editing the `FILTER_SERVER` configuration files to set the operating parameters of IPS; for example, the applications that will use IPS and setting the rule parameters for reporting events.

**Note:** The `FILTER_SERVER` process will not be started if the file `MULTINET:FILTER_SERVER_CONFIG.TXT` does not exist.

## Configuring Process-Specific Parameters

Logical names are used to set process-specific parameters for the `FILTER_SERVER` mailbox and some of the process-specific quotas for the `FILTER_SERVER` process. These logical names are:

```
MULTINET_FILTER_SERVER_TQELM  
MULTINET_FILTER_SERVER_ASTLM  
MULTINET_FILTER_SERVER_MBX_MSGS
```

## Determining the Correct `FILTER_SERVER` Process Quotas

It is important to correctly determine the correct process quotas for the `FILTER_SERVER` process. High-volume systems, for example, an E-mail server where SMTP may detect many anomalies, may log

large numbers of events in a short time. If the TQELM and ASTLM quotas for FILTER\_SERVER are too low, the FILTER\_SERVER process could enter MUTEX state and hang, preventing any events from being logged and possibly leading to other problems such as processes hanging while trying to log events.

The amount of additional TQELM quota in addition to the default value (specified via the PQL\_DTQELM SYSGEN parameter) required for the FILTER\_SERVER process can be calculated as follows:

- 1 for automated hourly reporting
- 1 for automated 24-hour maintenance
- 1 for each source address per rule per component for which an event has been received. These timers are used to clean up internal address structures and disappear after 24 hours of inactivity from that address.
- 1 for each non-empty event queue per source address per rule per component. These timers are used to delete aged events from the event queue.

Thus, the event frequency must be anticipated and the quotas adjusted appropriately for each installation. The hourly FILTER\_SERVER logs will be of use for determining traffic patterns.

The ASTLM quota tends to track the value for TQELM closely, but should have an additional 10% added to it for other uses.

Both the ASTLM and TQELM quotas are controlled by logical names described in the previous section. Both of the ASTLM and TQELM values default to 500.

## Determining the Correct FILTER\_SERVER Mailbox Size

In addition to setting the TQELM and ASTLM process quotas correctly, the size of the mailbox used for communication with the FILTER\_SERVER process must be correctly determined. Failure to do can result in events reported by instrumented components being lost. The mailbox is sized to handle 400 simultaneous event messages by default.

Once the mailbox size has been configured, either the system must be rebooted to allow the new mailbox size to be used (this is the preferred method), or the following procedure can be used to avoid a reboot in the near term:

1. Stop IPS (`MULTINET SET /IPS /STOP`).
2. Stop all applications using IPS (e.g., telnet sessions, ftp session, etc.).
3. Delete the old mailbox by running `MULTINET : DELMBX . EXE`.

4. Start IPS (`MULTINET SET /IPS /START`).
5. Start any other applications previously stopped.

## Filter Server Main Configuration

The filter server is configured using a main configuration file and per-component configuration files. The main configuration file is used to set overall configuration options for filter server operation, while the per-component configuration files contain configuration information for each instrumented component such as the ruleset to use, the prototype filter to be set, etc. Per-component configuration files are referenced by the main configuration file by using the `INCLUDE` keyword.

Sample configuration files are supplied in the MultiNet distribution and must be copied and modified as necessary to conform to the particular site's security profile and interface configuration. These files are copied to the `MULTINET` directory when MultiNet is installed. Once these have been copied and modified, the filter server configuration may be reloaded via the `MULTINET SET/IPS/RELOAD` command. The template files supplied are:

```
FILTER_SERVER_CONFIG.TEMPLATE
SSH_FILTER_CONFIG.TEMPLATE
IMAP_FILTER_CONFIG.TEMPLATE
POP3_FILTER_CONFIG.TEMPLATE
SNMP_FILTER_CONFIG.TEMPLATE
SMTP_FILTER_CONFIG.TEMPLATE
TELNET_FILTER_CONFIG.TEMPLATE
REXEC_FILTER_CONFIG.TEMPLATE
RSHELL_FILTER_CONFIG.TEMPLATE
RLOGIN_FILTER_CONFIG.TEMPLATE
```

The following table lists the main configuration file keywords. These are found in the file `MULTINET:FILTER_SERVER_CONFIG.TXT`:

Keyword	Default	Description
<code>BLOCK_AT_DESTINATION_PORT</code>	NO	<p>If set to YES, indicates that all filters generated by the filter server will be for a specific destination port (the equivalent of a filter line of "EQ &lt;portnumber&gt;"). The port number to be used is specified for each component in the per-component configuration file.</p> <p>If set NO, all filters generated by the filter server will deny access to all destination ports.</p>



DEBUG <i>value</i>	0	Indicates the amount of debug to output. Zero means no debug, while higher number mean more debug. This value should ordinarily never be set above 4 without direction from Process Software.
ENTERPRISE_STRING		Defines the location in the MIB tree that the trap used to send filter logging events via SNMP pertains to.
GENERIC_TRAP_ID		An integer representing the generic trap value when filter logging events are sent via SNMP.
INCLUDE <i>filename</i>		Specifies a per-component configuration file to load. Any number of INCLUDE statements may occur in the main configuration file.
LOG_TO_LOGFILE	NO	If YES, information log messages are sent to the log file specified by the logfile keyword.
LOG_TO_OPCOM	NO	If YES, informational messages are reported via OPCOM.
LOG_TO_SNMP	NO	If YES, informational messages are reported via an SNMP trap.
LOGFILE		Specifies the log file used when the log_to_logfile keyword is specified.
OPCOM_TARGET	NETWORK, SECURITY	Specifies the list of operator types to which events are written when the keyword LOG_TO_OPCOM is set. This is a comma-separated list, and may contain any of the values which are valid for the VMS REPLY/ENABLE command.
SPECIFIC_TRAP_ID		An integer representing the specific trap value when filter logging events are sent via SNMP.

# Filter Server Per-Component Configuration File

The per-component configuration files are loaded using the `INCLUDE` keyword in the main filter server configuration file. Each of these configuration files have the following format. The definition must begin with a `COMPONENT` keyword. Comments lines begin with a `#` character.

```
COMPONENT component-name
    DESTINATION_ADDRESS
    EXCLUDE_ADDRESS
    DESTINATION_PORT
    PROTO_FILTER
RULE rulename
    DESTINATION_ADDRESS
    DESTINATION_PORT
    MAX_COUNT
    DELTA_TIME
    FILTER_DURATIONS
RULE rulename
    MAX_COUNT
    DELTA_TIME
    FILTER_DURATIONS
```

Each component may have as many rules defined for it as are appropriate for the component. However, the more rules defined for a component, the more complex it may be to instrument the component to actually report those rules. All entries in configuration files are not case-sensitive.

The following table shows the keywords for a per-component configuration file:

Keyword	Scope	Description
<code>COMPONENT <i>component-name</i></code>	component	Name of the component to which this applies (e.g., SSH).
<code>DELTA_TIME <i>time</i></code>	rule	Time, in seconds, where if <code>max_count</code> events are received for a rule from the same address, that will cause a filter to be set for that address  This is also the time for aging events. If the age of an event exceeds <code>delta_time</code> seconds, the event is dropped from the event list.
<code>DESTINATION_ADDRESS <i>address</i></code>	component or rule	Destination IP address (the MultiNet interface address) in CIDR format to check. This may be in IPv4 or IPv6 format.

		<p>If <code>destination_address</code> occurs before the first rule in the per-component configuration file, it will be used as a default for any rule for the component that doesn't have a destination address specified.</p> <p>Note: multiple <code>destination_address</code> lines may be specified at the component level if all the interfaces specified have the same filtering criteria.</p>
<code>DESTINATION_PORT</code> <i>port</i>	component or rule	Optional destination port. This will only be effective if the keyword <code>BLOCK_AT_DESTINATION_PORT</code> is set in the main configuration file.
<code>EXCLUDE_ADDRESS</code> <i>address</i>	component or rule	<p>A source address/mask in CIDR format from which events are ignored. This allows, for example, events from an internal network to be ignored while counting events from external networks.</p> <p>Multiple <code>EXCLUDE_ADDRESS</code> lines may be specified for each rule.</p>
<code>FILTER_DURATIONS</code> <i>list</i>	rule	List of durations for filters. This is a comma-delimited list of up to five filter durations, and it must be terminated with a -1.
<code>MAX_COUNT</code> <i>count</i>	rule	Maximum number of events from the same address for a specific rule over <code>DELTA_TIME</code> seconds that will trigger a filter.
<code>PROTO_FILTER</code>	component	This is a prototype filter to be used to build the filter set against an interface when a filter is triggered. The format of this filter is the same used in a filter file.
<code>RULE</code> <i>rulename</i>	component	The user-defined name for a rule.

# Sample Main Configuration File

```
#####  
#  
#           FILTER_SERVER_CONFIG.TEMPLATE  
#  
#####  
#  
#           The following parameter determines the level of debug information  
#           written to the debug log file.  This should normally be set to a  
#           value of 2 or less, and shouldn't be set above 4 without the  
#           recommendation of Process Software, as higher debug levels will  
#           negatively impact the filter server (and possibly, system)  
#           performance.  The debug messages will be found in the file  
#           MULTINET:FILTER_SERVER.OUT.  
#  
debug          4  
#  
#           The following parameters define the logging locations.  Note  
#           that debug messages are not written to the logging locations.  
#  
#           The first two parameters are used when logging to a log file.  
#  
logfile        multinet:filter_logfile.log  
log_to_logfile      yes  
#  
#           The next parameter is used when logging to OPCOM.  
#  
log_to_opcom      yes  
opcom_target      NETWORK,SECURITY,OPER3  
#  
#           The next parameters are used when logging via SNMP.  Details  
#           on the values for enterprise_string, generic_trap_id and  
#           specific_trap_id can be found in chapter 23 of the MultiNet  
#           Administrators Guide.  
#  
log_to_snmp       no  
# enterprise_string  
# generic_trap_id  
# specific_trap_id  
#  
#           The following parameter determines how filters are created.  
If  
#           set to YES, then the destination port field is added to the  
filter
```

```

#           (e.g., "192.168.0.11/32 eq 22").  If set to NO, then no source
#           port field is added, which will cause the filter to block all
#           traffic of the specified protocol from the source address.  If
#           not set, default is NO.
#
# block_at_destination_port yes
#
#=====
#
#           The following lines define the individual configuration files
#           for each configured component that uses the filter server
#
#=====
#
include multinet:ftp_filter_config.txt
include multinet:imap_filter_config.txt
include multinet:pop3_filter_config.txt
include multinet:smtp_filter_config.txt
include multinet:snmp_filter_config.txt
include multinet:ssh_filter_config.txt
#

```

For this configuration:

- Debug will be reported at level 4 (this produces fairly detailed information, normally useful only by Process Software when debugging a problem).
- Log messages will be logged to `MULTINET:FILTER_LOGFILE.LOG` and `OPCOM`.
- When filters are logged, the destination port specified in the per-configuration files will be used.
- Per-component configuration files for the MultiNet FTP, IMAP, POP3, SMTP, SNMP and SSH servers will be loaded.

## Sample Component Configuration File

The following is a configuration file for the SSH component:

```

component ssh
  proto_filter "deny tcp 192.168.0.100/32 192.168.0.11/32 log"
  destination_address 192.168.0.16/32
  exclude_address 192.168.0/24
  destination_port 22
  rule    ssh_bogus_id
          max_count      10
          delta_time     90

```

```

rule      filter_durations  300,600,1800,3600,-1
         ssh_authfailed
         max_count          10
         delta_time         90
rule      filter_durations  300,600,1800,3600,-1
         ssh_authfailed
         destination_address 192.168.10.2/16
         max_count          10
         delta_time         90
rule      filter_durations  300,600,1800,3600,-1
         ssh_userauth
         max_count          10
         delta_time         90
rule      filter_durations  300,600,1800,3600,-1
         ssh_invaliduser
         max_count          10
         delta_time         90
         filter_durations  300,600,1800,3600,-1

```

For component SSH, a `deny tcp` filter will be used. The source address/mask and destination address/mask parts of the prototype filter are ignored and are overwritten by the actual data specified by the source information gathered from the event that triggered the filter, and by the destination address/mask/port information specified by the corresponding keywords in this file. Events from the 192.168.0 network are all excluded from being counted.

To examine the first three rules specified above:

The rule is `ssh_bogus_id`. Since no address or mask is specified for this rule, it will use the default destination address of 192.168.0.16 and mask of 255.255.255.255 specified at the beginning of the component configuration. The rule states that if 10 events from the same source address are seen over 90 seconds, a filter is created using the `proto_filter` specified above. The first filter is 5 minutes long, the second, 10 minutes, and so on, until at the 5th time, a permanent filter is put in place for the address and interface that is causing the problem.

The second rule is `ssh_authfailed`, and applies to events received as a result of connections on the interface with the default address of 192.168.0.16 and mask of 255.255.255.255, respectively.

The third rule is also `ssh_authfailed`, but applies to events received a result of connections on the interface with the address 192.168.10.2, using a mask of 255.255.0.0. The `max_count` and `delta_time` parameters are different for this interface than for the previous `ssh_authfailed` rule in the system.

The remaining rules for this component will use the default address 192.168.0.16 and mask of 255.255.255.255.

**Note:** If a rule specifies a destination address for an interface which does not currently exist, events for that interface will be dropped until the interface becomes available.

If your system has multiple interfaces (for example, SE0, SE1 and PD0), you must specify all interfaces in the same config file. For each rule in the config file, you must supply a separate section for each destination address (i.e., interface). The `component` keyword may occur exactly once in the configuration file. The following example shows a config file for component `ftp` for 5 interfaces (SE0, SE1, PD0, PD1, PD2):

```
#####  
#  
#      FTP_FILTER_CONFIG.TXT  
#  
#      Filter server configuration file for the FTP component.  
#  
#####  
component ftp  
    proto_filter "deny tcp 192.168.0.100/32 192.168.0.1/24 log"  
    destination_port 21  
#  
# For SE0 and SE1  
#  
    destination_address 192.168.0.29/32  
    destination_address 192.168.0.25/32  
    rule    ftp_invaliduser  
            max_count      10  
            delta_time     300  
            filter_durations 300,600,1800,3600,-1  
            destination_address 192.168.0.29/32  
    rule    ftp_userauth  
            max_count      21  
            delta_time     180  
            filter_durations 300,600,1800,3600,-1  
            destination_address 192.168.0.29/32  
    rule    ftp_authfailed  
            max_count      21  
            delta_time     90  
            filter_durations 300,600,1800,3600,-1  
            destination_address 192.168.0.29/32  
    rule    ftp_timeout  
            max_count      21  
            delta_time     90  
            filter_durations 300,600,1800,3600,-1
```

```

destination_address 192.168.0.29/32
#
# For PD0
#
rule ftp_invaliduser
max_count 10
delta_time 300
filter_durations 300,600,1800,3600,-1
destination_address 192.168.0.28/32
destination_port 1521
rule ftp_userauth
max_count 21
delta_time 180
filter_durations 300,600,1800,3600,-1
destination_address 192.168.0.28/32
destination_port 1521
rule ftp_authfailed
max_count 21
delta_time 90
filter_durations 300,600,1800,3600,-1
destination_address 192.168.0.28/32
destination_port 1521
rule ftp_timeout
max_count 21
delta_time 90
filter_durations 300,600,1800,3600,-1
destination_address 192.168.0.28/32
destination_port 1521
#
# For PD1
#
rule ftp_invaliduser
max_count 10
delta_time 300
filter_durations 300,600,1800,3600,-1
destination_address 192.168.0.27/32
exclude_address 192.168.0.0/24
rule ftp_userauth
max_count 21
delta_time 180
filter_durations 300,600,1800,3600,-1
destination_address 192.168.0.27/32
exclude_address 192.168.0.0/24
rule ftp_authfailed
max_count 21
delta_time 90
filter_durations 300,600,1800,3600,-1
destination_address 192.168.0.27/32
exclude_address 192.168.0.0/24
rule ftp_timeout
max_count 21
delta_time 90

```



```

        filter_durations 300,600,1800,3600,-1
        destination_address 192.168.0.27/32
        exclude_address 192.168.0.0/24
#
# For PD2
#
    rule ftp_invaliduser
        max_count 10
        delta_time 300
        filter_durations 300,600,1800,3600,-1
        destination_address 192.168.0.21/32
    rule ftp_userauth
        max_count 21
        delta_time 180
        filter_durations 300,600,1800,3600,-1
        destination_address 192.168.0.21/32
    rule ftp_authfailed
        max_count 21
        delta_time 90
        filter_durations 300,600,1800,3600,-1
        destination_address 192.168.0.21/32
    rule ftp_timeout
        max_count 21
        delta_time 90
        filter_durations 300,600,1800,3600,-1
        destination_address 192.168.0.21/32

```

The above example shows some configuration options for the system with 5 interfaces. Specifically:

- Interfaces SE0 and SE1 will use identical rules, because they didn't specify destination addresses within their rulesets and the destination addresses for SE0 and SE1 were specified at the component level. All other interface rules specified their own destination addresses at the rule level, so they will use specific rules for those specific addresses.
- A default port of 21 has been specified for all interfaces. However, interface PD0 has specified a port of 1521, so that port will be used for PD0 only. All other interfaces will use the default port of 21.
- Interface PD1 has an `exclude_address` specified for net 192.168.0.0/24. All events for PD1 that originated from that source net will be excluded from being counted by IPS. All other interfaces will count events from that network.

# Configuring IPS for Paired Network Interfaces

To configure IPS for a paired network interface environment where multiple interfaces are treated as a common link set, the rules are fairly simple.

- Each physical and pseudo interface must be specified in the configuration files via `destination_address` rules for each interface.
- All physical interfaces are treated equally. When an event is logged for any interface in the set, it's as if it was logged against each interface in the set. Thus, when a filter is set on any interface in the set, the same filter is set on all physical interfaces in the set.
- Filters are set only on the physical interfaces. Since pseudo devices (PDnnn) are not true interfaces, they cannot have filters set on them.
- When a filter is created as a result of events coming in via a pseudo device, the destination address shown in the filter (using the `MULTINET SHOW INTERFACE /FILTER` command) will show the destination address for the pseudo device.

When `MULTINET SET INTERFACE` is used to perform any of the following tasks:

- Create a paired network interface set via `SET INTERFACE /COMMON_LINK`
- Start an interface via `SET INTERFACE/UP`

IPS is notified of the change being made. This allows the `FILTER_SERVER` process to reevaluate all interfaces it knows about, so it can determine if modifications must be made to paired network interface sets about which it currently knows.

The following example shows a configuration for the SSH component for a paired network interface configuration that consists of SE0, SE1, and PD0 where PD0's physical interface is SE1:

```
component ssh
  proto_filter "deny tcp 192.168.0.100/32 192.168.0.11/24 log"
  #
  # SE0's address
  #
  destination_address 192.168.0.70/32
  #
  # SE1's address
  #
  destination_address 192.168.0.71/32
  #
  # PD0's address
  #
  destination_address 192.168.0.72/32
  #
```

```

destination_port 22
rule  ssh_bogus_id
      max_count      10
      delta_time     90
      filter_durations 300,600,1800,3600,-1
rule  ssh_authfailed
      max_count      10
      delta_time     90
      filter_durations 300,600,1800,3600,-1
rule  ssh_authfailed
      destination_address 192.168.10.2/32
      max_count      10
      delta_time     90
      filter_durations 300,600,1800,3600,-1
rule  ssh_userauth
      max_count      10
      delta_time     90
      filter_durations 300,600,1800,3600,-1
rule  ssh_invaliduser
      max_count      10
      delta_time     90
      filter_durations 300,600,1800,3600,-1

```

Using the above configuration, the next item illustrates a filter being set due to events that occurred on line PD0:

```

BIGBOOTE_$
%%%%%%%%%% OPCOM 29-OCT-2019 13:00:55.77 %%%%%%%%%%% (from node BOS1
at 29-OCT-2019 13:00:59.12)
Message from user JOHNDOE on BOS1
FILTER_SERVER: Filter queued on SE0 (192.168.0.70/32) at 29-OCT-2019
13:00:59.12
      Component: ssh, Rule: ssh_bogus_id
deny      tcp      192.168.0.11/32
           192.168.0.72/32      eq 22
           FLTSVR,LOG
           START: 29-OCT-2019 13:00:59.12  END: 29-OCT-2019
14:00:59.12

BIGBOOTE_$
%%%%%%%%%% OPCOM 29-OCT-2019 13:00:55.80 %%%%%%%%%%% (from node BOS1
at 29-OCT-2019 13:00:59.15)
Message from user JOHNDOE on BOS1
FILTER_SERVER: Filter queued on SE1 (192.168.0.71/32) at 29-OCT-2019
13:00:59.15
      Component: ssh, Rule: ssh_bogus_id
deny      tcp      192.168.0.11/32
           192.168.0.72/32      eq 22
           FLTSVR,LOG

```

```
START: 29-OCT-2019 13:00:59.15  END: 29-OCT-2019
14:00:59.15
BIGBOOTE_ $
```

Note some things illustrated above:

- Each physical address (SE0 and SE1) had a filter set on it.
- No filter was set on interface PD0 because it is a pseudo interface.
- The destination address for each event is that of interface PD0, since that was the source of the events that caused the filters to be set.

## Filter Reporting via OPCOM and Log File

When a filter is set for an address/rule/destination/component, an informational message will appear either in OPCOM (if LOG\_TO\_OPCOM is set) or in the log file (if LOG\_TO\_LOGFILE is set). The following message illustrates an OPCOM message, but the message to a log file will have the same format.

```
TWEET_ $
%%%%%%%%% OPCOM 16-MAY-2019 10:33:19.74 %%%%%%%%%%
Message from user SYSTEM on TWEET
FILTER_SERVER: Filter queued on se0 (192.168.0.16) at 16-MAY-2019
10:33:19.74
          Component: ssh, Rule: ssh_bogus_id
deny      tcp      192.168.0.11/32
           192.168.0.0/24      eq 22
           FLTSVR, LOG
           START: 16-MAY-2019 10:33:19  END: 16-MAY-2019 10:38:19
TWEET_ $
```

This message is in essentially the same format as that when a MULTINET SHOW/INTERFACE/FILTER command is performed:

```
TWEET_ $ mu show/interface se0/filter
Device se0: flags=8863<UP,BROADCAST,NOTRAILERS,RUNNING,MULTICAST,D2>
           VMS Device = EWA0
           IP Address = 192.168.0.16
           No common links defined

MultiNet Packet Filter List for se0:
```

```
Logging is disabled
```

```

Action      Proto      Hits      Source Address / Port
-----      -
deny        tcp         0         192.168.0.11/32
           tcp         0         192.168.0.0/24  eq 22
           tcp         0         FLTSVR,LOG
           START: 16-MAY-2019 10:33:19  END: 16-MAY-2019 10:38:19

permit      ip        13484     0.0.0.0/0
           ip        13484     0.0.0.0/0
           ip        13484     FLTSVR

Average 0 bytes out, 0 bytes in per second
Average 0 packets out, 0 packets in per second

TWEET_$
```

Note the second filter (the `permit ip` filter) that is shown. If there are currently no filters set for an interface when the filter server determines it needs to set a filter, it will add an explicit `permit ip` filter. This is done because the existence of any filter in a list of filters causes MultiNet to act as if a `deny everything` filter terminates the list. The `permit ip` filter will essentially prevent that problem from happening.

## Filter Reporting via SNMP

When logging a filter via SNMP, the configuration keywords `ENTERPRISE_STRING`, `GENERIC_TRAP_ID`, and `SPECIFIC_TRAP_ID` must be specified (as well as the keyword `LOG_TO_SNMP`). In addition, the SNMP configuration file must be properly set up on the MultiNet system as specified in chapter 23 of the *MultiNet Installation and Administrator's Guide*.

When a filter is logged, the following fields will be reported:

```
FILTER_SERVER: Filter queued on interface (address) at time
COMPONENT=component-name
RULE=rulename
ACTION=actionname      (e.g., "deny")
PROTOCOL=protocol      (e.g., "TCP")
SOURCE=source address in CIDR format
SOURCE_PORT=operator port      (e.g., "EQ 22")
DESTINATION=destination address in CIDR format
DEST_PORT=operator port      (e.g., "EQ 22")
START=VMS absolute time
```

# Correcting a Filter List

If a filter is inadvertently created by the filter server, the system manager should first correct the configuration problem (if one exists) that allowed the filter to be incorrectly set. Then, the system manager may retrieve the current list of filters in “manual filter form” that can be edited then reloaded onto the interface. The list is retrieved via the MU SHOW/INTERFACE/EXTRACT\_FILTER command. For example:

```
TWEET_$ mu show/int se0/filt
Device se0: flags=8863<UP,BROADCAST,NOTRAILERS,RUNNING,MULTICAST,D2>
          VMS Device = EWA0
          IP Address = 192.168.0.16
          No common links defined

MultiNet Packet Filter List for se0:

Logging is disabled

Action      Proto      Hits      Source Address / Port
-----      -
deny        tcp         0      192.168.0.11/32
           192.168.0.0/24 eq 22
           FLTSVR,LOG
           START: 16-MAY-2019 10:33:19  END: 16-MAY-2019 10:38:19

deny        tcp         15      192.168.0.38/32
           192.168.0.11/24 eq 22

permit      ip        13484    0.0.0.0/0
           0.0.0.0/0
           FLTSVR

          Average 0 bytes out, 0 bytes in per second
          Average 0 packets out, 0 packets in per second
TWEET_$ mu show/interface se0/extract_filter=filter.txt
TWEET_$ type filter.txt
#
#  FILTER.TXT
#
#  Generated 16-MAY-2019 10:51:31
#
#=====
deny tcp 192.168.0.100/32 192.168.0.11/24 eq 22 start "16-MAY-2019 10:33:19"
end "16-MAY-2019 10:38:19"LOG
deny tcp 192.168.0/32.192.168.0.11/24
```

```

permit ip 0.0.0.0/32 0.0.0.0/32
TWEET_$ <edit to remove the first (filter) line>
TWEET_$ mu set/interface se0/filter=filter.dat
TWEET_$ mu show/interface se0/filt
Device se0: flags=8863<UP,BROADCAST,NOTRAILERS,RUNNING,MULTICAST,D2>
          VMS Device = EWA0
          IP Address = 192.168.0.16
          No common links defined

MultiNet Packet Filter List for se0:

Logging is disabled

Action      Proto      Hits      Source Address / Port
-----
deny        tcp        15        192.168.0.38/32
           tcp        15        192.168.011/24 eq 22

permit      ip         13484     0.0.0.0/0
           ip         13484     0.0.0.0/0

          Average 0 bytes out, 0 bytes in per second
          Average 0 packets out, 0 packets in per second
TWEET_$

```

## Configuring PMDF to use IPS on MultiNet

The IMAP, POP3 and SMTP servers referred to in the configuration template files above refer to the MultiNet servers only. Beginning with PMDF V6.4, PMDF has been instrumented to use IPS. The following PMDF template files are available in the PMDF\_TABLE directory:

```

FILTER_SERVER_PMDF_IMAP.TEMPLATE
FILTER_SERVER_PMDF_POP3.TEMPLATE
FILTER_SERVER_PMDF_SMTP.TEMPLATE

```

These files should be copied to MULTINET:\* .TXT and modified as appropriate for your installation. Edit MULTINET:FILTER\_SERVER\_CONFIG.TXT to add INCLUDE lines for these component files, and comment out the INCLUDE lines for the standard MultiNet IMAP, POP and SMTP files.

Next, make sure the appropriate PMDF images are installed. The legacy IMAP and POP servers (PMDF\_EXE:IMAPD.EXE and PMDF\_EXE:POP3D.EXE) are already installed. The msgstore IMAP and POP servers (PMDF\_EXE:IMAP\_SERVER.EXE, PMDF\_EXE:POP\_SERVER.EXE), as well as the SMTP server (PMDF\_EXE:TCP\_SMTP\_SERVER.EXE) are not installed, so they will need to be

added to your `PMDF_COM:PMDF_SITE_STARTUP.COM` file if your PMDF installation uses them. These must all be installed using the `/OPEN` qualifier.

At this point, define the logical name `PMDF_DO_FILTER_SERVER` to 1, using the `/SYSTEM` qualifier (this can be put in `PMDF_COM:PMDF_SITE_STARTUP.COM` as well).

Finally, restart IPS via the `MULTINET SET/IPS/RESTART` command.

## Controlling the Filter Server

The filter server is started at system startup time. However, it can be controlled using the `MULTINET SET/IPS` command. The valid commands and their uses are:

Command	Description
<code>/DEBUG_LEVEL=level</code>	Change the debug level for the server. See the description for the <code>DEBUG</code> main configuration keyword.
<code>/CLEAR_FILTERS</code>	Causes the <code>FILTER_SERVER</code> process to remove all filters set by IPS on all interfaces configured for IPS. This may be used with <code>SET /IPS /START</code> and <code>SET /IPS/RESTART</code> , or may be used by itself with <code>SET /IPS/CLEAR_FILTERS</code> . When used by itself this causes a running IPS subsystem to remove the IPS filters and reset the event count information for the source address associated with each filter being removed.
<code>/RELOAD</code>	Re-read and parse the configuration files. Note that this will not wipe out existing event and rule information; it will simply update it so no potential filter information will be lost.
<code>/RESTART</code>	Stop and restart the filter server. All existing event and rule information will be lost and reloaded from the configuration files.
<code>/START</code>	Start the filter server if it's not already running.
<code>/STOP</code>	Stop the filter server from running. All existing event and rule information will be lost.



The current configuration of the filter server may also be displayed using the MULTINET SHOW/IPS /CONFIG=*filename* command. For example:

```
$ multinet show/ips/config=server_stats.out
$ type server_stats.out
Filter server snapshot

Debug level 6
Block at destination port or system: PORT
Log to: OPCOM   SNMP trap
Component  ssh
  Rule ssh_bogus_id
    dest address:      192.168.0.16/32
    interface:         se0
    max event count:   10
    delta time:        0 00:01:30.00
    filter durations:  300 600 1800 3600 -1
    Address 192.168.0.11/32
      number of still-queued events: 1
      number of filters created:     0
      Address entry to be deleted:    N/A
    Event
      event time: 29-APR-2019 10:00:12.41
      expires:   29-APR-2019 10:01:42.41
  Rule ssh_authfailed
    dest address:      192.168.0.16/32
    interface:         se0
    max event count:   10
    delta time:        0 00:01:30.00
    filter durations:  300 600 1800 3600 -1
  Rule ssh_userauth
    dest address:      192.168.0.16
    interface:         se0
    max event count:   10
    delta time:        0 00:01:30.00
    filter durations:  300 600 1800 3600 -1
  Rule ssh_invaliduser
    dest address:      192.168.0.16/32
    interface:         se0
    max event count:   10
    delta time:        0 00:01:30.00
    filter durations:  300 600 1800 3600 -1
  Rule ssh_invalid_id_msg
    dest address:      192.168.0.16/32
    interface:         se0
    max event count:   5
    delta time:        0 00:02:00.00
    filter durations:  300 600 1800 3600 -1
```

# Filter Server Files

The following files are associated with the filter server:

`MULTINET:FILTER_SERVER_HOURLY_LOG.yyyymmdd`

This file is an hourly activity log for the filter server. The file extension changes daily at midnight to reflect the current day. What follows is a sample log segment for one hour:

```
Filter server hourly snapshot for hour 2 of 05/18/2019

Component  ssh

  Rule ssh_bogus_id
    number of hits 0

  Rule ssh_authfailed
    number of hits 0

  Rule ssh_userauth
    number of hits 0

  Rule ssh_invaliduser
    number of hits 2

  Address 192.168.0.10/32
    number of still-queued events: 0
    number of all events: 0
    number of filters created: 1
    Address entry to be deleted: 18-MAY-2019 05:55:45.45

  Address 192.168.0.204
    number of still-queued events: 0
    number of all events: 2
    number of filters created: 0
    Address entry to be deleted: 18-MAY-2019 06:22:03.97
```

This log is showing that during the hour 01:00-02:00, 2 different source addresses were being tracked by the filter server.

The first address (192.168.0.10) had a filter created sometime in the last 4 hours (the time it takes an address to have no activity before its records are deleted by the filter server). The log indicates the address entry is to be deleted in 3 hours if there is no more activity; therefore, the filter was actually set in the previous hour (looking at the previous hour's entry in the log file will confirm this).

The second address (192.168.0.204) had 2 events during the hour that never triggered a filter and were deleted after they aged. This address entry is scheduled to be deleted in 4 hours if there's no more activity for it.

MULTINET:FILTER\_SERVER\_CONFIG.TXT

This is the main filter server configuration file. Optionally, the server will use the logical name `FILTER_SERVER_CONFIG` to determine the name of the main configuration file.

MULTINET:FILTER\_SERVER.OUT

This file contains any output resulting from starting the filter server (e.g., the output from any DCL commands executed to start it) and all debug messages.

# Instrumenting a User-Written Application with IPS

When instrumenting an application (aka, a component), there are several steps to be followed:

- The user determines the component-specific parameters. These include:
  - The prototype filter to be used when a filter is created. This is the same format as that used when using a filter file. All filter features are supported, with the exception of the `ESTABLISHED` and `REPEATING` keywords. Note that the source address/mask/port and destination address/mask/port fields of the filter will be overwritten during creation of the filter, according to the other parameters set in the configuration file.
  - Whether the filters created will block at the destination port or simply block all traffic from the source system (the `BLOCK_AT_DESTINATION_PORT` keyword).
  - The logging to be used.
- The user determines the ruleset:
  - The user determines what rules are to be supported. There's no limit on the number of rules the filter server can maintain; the limit is really on how complex you want to make the component.
- For each rule, you need to determine:
  - The name of the rule. This string (maximum length 35 characters) will be used by the filter server and by the call to the filter server API call `send_filter_event`.
  - The number of events/unit time that will trigger a filter (the `MAX_COUNT` and `DELTA_TIME` fields).
  - The duration(s) of a filter. Up to 5 may be chosen, and the list must end with `-1`.
- The user creates the component-specific configuration file, then adds a reference to it via the `INCLUDE` keyword in the main filter server configuration file. At this point, the filter server can be made aware of this new (or updated) configuration by using the `MULTINET SET /IPS/RELOAD` command.

**Note:** The filter server configuration may be reloaded multiple times without causing problems for the filter server.

## Filter Server API

There are two calls available in the filter server API. The function prototypes are defined in `MULTINET_COMMON_ROOT:[MULTINET.INCLUDE]FILTER_SERVER_API.H`. The first call is used to register with the filter server:

```
int filter_server_register(char *component, 0, 0)
```

where `component` is the name of the component.

The remaining two arguments are there for future expansion, and are ignored, but must be specified.

The return values from this function are 1 (success) or 0 (an error occurred; most likely, this is because the filter server isn't running). Normally this function is called once when the first event is logged. However, if an error is returned, it may be called again when additional events are logged.

**Note:** The application that's registering **MUST** be an installed image, using `/OPEN` or `/SHARED`. It doesn't need to be installed with privileges. This is an attempt to help cut down on bogus applications registering with the server; it takes a conscious effort - and privileges - by the system manager to do this and therefore, to control this.

The next function is used to format and send events to the filter server:

```
int send_filter_event(char *rule,  
                    char *source_address,  
                    u_short source_port,  
                    char *dest_address)
```

where:

`rule` is the name of the rule to be enforced. This must correspond to a `rule` keyword specified in the per-component configuration file for the component. If a match cannot be made, the event will be ignored by the filter server.

`source_address` is the address of the system that caused the event to be logged (e.g., "192.168.0.1"). This may be in IPv4 or IPv6 format, but must be of the same address family as that of the `destination_address` specified for the component in the per-component configuration file. Note that this is an address only. Do not specify address mask bits (e.g., "192.168.0.1/24") with it.

`source_port` is the source port on the originating system.

`dest_address` is the destination address of the socket used to communicate to `source_address`. This information may be obtained by performing a `getsockname` function on the socket. Note that this is an address only. Do not specify address mask bits (e.g., "192.168.0.11/24") with it.

To include these routines in your application, link using the library

`MULTINET_COMMON_ROOT:[MULTINET.LIBRARY]FILTER_SERVER_API.OLB.`

The following is an example of code used to send events to the filter server:

```
void ssh_send_filter_event(int code, char *addr, int port, char *dest_addr)
{
    char *rule;
    static int filter_server = -1;

    if (filter_server == -1)
        filter_server = filter_server_register("SSH", 0, 0);

    if (!filter_server)
        return;

    switch(code)
    {
        case LGI$_NOSUCHUSER:
        case LGI$_NOTVALID:
            rule = "SSH_INVALIDUSER";
            break;

        case LGI$_USERAUTH:
            rule = "SSH_USERAUTH";
            break;

        case LGI$_DISUSER:
        case LGI$_ACNTEXPIR:
        case LGI$_RESTRICT:
        case LGI$_INVPWD:
        case LGI$_PWDEXPIR:
            rule = "SSH_AUTHFAILED";
```

```
        break;

    default:
        printf("Unrecognized status code %d", code);
        return;
}

send_filter_event(rule, addr, (unsigned short)port, dest_addr);
}
```

# 31. Using MultiNet for an OpenVMS Cluster Interconnect

MultiNet can be used to provide transport services for an OpenVMS IP cluster on Alpha and Integrity systems running OpenVMS V8.4 or higher. Users should first follow the directions in the *Cluster over IP* section in the OpenVMS Guidelines for Cluster Configurations manual. VMS configuration is done with `SYS$MANAGER:CLUSTER_CONFIG_LAN.COM`.

When MultiNet is installed on a system with IP clustering enabled, the MultiNet images will be copied to the appropriate system directories such that upon reboot the MultiNet images will be used instead of the TCP/IP Services images. MultiNet determines that the system is configured for IP clusters by checking the version of OpenVMS (8.4 or later) and the `SYSGEN` parameter `NISCS_USE_UDP`. MultiNet will still need to be configured as the configuration is not copied from the TCP/IP Services files. The TCP/IP Services images are duplicated in the system directories with the extension `.TCPIP_EXE` and `.TCPIP_STB` before copying the MultiNet images. This allows for returning to the TCP/IP Services images should there be a problem with MultiNet.

The following line will be displayed during installation if the system is configured for IP Clustering:

```
%MULTINET-I-IPCLUSTER, Updating IP Cluster images
```

On systems that do not have IP clustering enabled when MultiNet is installed follow the steps below.

1. Configure TCP/IP Services and IP clustering as documented in the *OpenVMS Guidelines for Cluster Configurations* manual. This will create the following files:

```
SYS$SYSROOT:[SYSEXE]TCPIP$CLUSTER.DAT  
SYS$SYSTEM:PE$IP_CONFIG.DAT
```

2. Execute `MULTINET:SET_MULTINET_IP_CLUSTER.COM` with the parameter `INITIAL` to enter the MultiNet files in the correct directories for VMS to find at boot time. This procedure will preserve the current TCP/IP Services image and copy the MultiNet images to the system directories with the appropriate names. The images are copied to the specific directories, not the common directories, so it will be necessary to manually move the images or do this on each system in the IP cluster that uses this boot disk. The following text will be displayed:

```
$ @multinet:set multinet ip cluster initial  
Saving HP's TCPIP$INTERNET_SERVICES.EXE as
```

```
SYS$LOADABLE_IMAGES:TCPIP$INTERNET_SERVICES.TCPIP_EXE
Updating sys$loadable_images:tcpip$internet_Services.exe with current
MultiNet image
Saving HP's TCPIP$BGDRIVER.EXE as
SYS$LOADABLE_IMAGES:TCPIP$BGDRIVER.TCPIP_EXE
updating sys$loadable_images:tcpip$bgdriver.exe with current MultiNet image
```

3. Use `SYSGEN` to verify that the system parameter `NISCS_USE_UDP` is set to 1 in the `CURRENT` set.
4. Reboot the system. The system boot process will say that it is loading TCP/IP Services, but there will also be a MultiNet initialization message on the console. After the reboot is complete use the standard MultiNet startup procedure to finish starting MultiNet; this will install the remaining network communication pseudo-devices, define logicals, install images and start the MultiNet master server and necessary ancillary processes for full network functionality.

## Troubleshooting

The following set of commands will verify that the MultiNet configuration and the IP cluster configuration agree. Any differences encountered will be displayed.

```
$ MULTINET CONFIGURE/NETWORK
MultiNet Network Configuration Utility 5.6
[Reading in configuration from MULTINET:NETWORK_DEVICES.CONFIGURATION]
NET-CONFIG> CHECK
```

If the MultiNet `KRNNOTFOUND` failed to locate MultiNet kernel message is displayed while attempting to start MultiNet and the BG device exists, then the most likely problem is that TCP/IP Services is being used instead of MultiNet. Use the MultiNet command procedure `MULTINET:SET_MULTINET_IP_CLUSTER.COM` to make sure that the MultiNet files are in the correct places.



# 32. Configuring Precision Time Protocol (PTP)

The Precision Time Protocol (PTP) service is an implementation of PTPv2 (IEEE 1588-2008). PTP is used to synchronize when clocks advance (tick) on a LAN computer network. It was designed primarily for instrumentation and control systems. On a local area network PTP is capable of synchronizing the clocks to an accuracy in the sub-microsecond range. In contrast, NTP is capable of accuracy in the tens of microseconds range. This higher degree of accuracy is achieved by using the time that the packet is received by the IP stack as part of the calculation to determine the difference between clocks. This means that the time between when the packet is received and when it is processed by the user mode application (PTP) can be included in the calculations when making adjustments to the skew between clocks. PTP is available for Alpha and Integrity systems running OpenVMS V7 and later.

When a PTP implementation starts up it is in the LISTEN state and it waits for an announcement from other systems. If no announcement happens within the `ptpengine:Announce_Receipt_Timeout`, then the system may move to MASTER state if it is configured to allow this. When in MASTER state it will announce its clock and the quality for other systems to observe. A system operating in LISTEN or SLAVE state will use the announced clock quality to choose the best master clock (BMC) from the announcements received. A system that is announcing a clock will also listen for other announcements and will stop making announcements and use another system as its master clock if a better one is detected. The quality of the clock is determined by the time source, how far it is removed from the source and the configuration parameters. The system with the reference clock is referred to as the Grand Master (GM). A potential master that has stopped making announcements will resume announcements when it stops receiving announcements from better masters.

UDP multicast is generally used on the local network to find other clocks, determine the best source and synchronize clocks. Currently PTP uses IPv4 only.

It is reasonable to run both NTP and PTP on a system, with NTP set as the `ptpengine:ptp_timesource`. This is reflected in the time source portion of the packets to let other systems know the quality of the clock. If the PTP configuration has `ntpengine:enabled = Y`, and the NTP configuration has `enable mode7` in the configuration PTP will attempt to disable NTP. NTP can also be set to be a failover mechanism for when a PTP master cannot be found.

PTP determines the offset from UTC on startup from the VMS system logical `SYS$TIMEZONE_DIFFERENTIAL` and uses a VMS system lock to synchronize with NTP for when

the TDF changes due to entering or leaving day light saving time. The PTP implementation does NOT have any code to change the VMS system logicals or offset when the system changes between standard time and day light saving time. Systems that observe day light saving time should use NTP to control that.

## Comparing PTP and NTP

PTP uses a significantly higher amount of CPU time and does a lot more I/O than NTP. The goal of PTP is to keep the clock synchronized with the selected grand master. The goal of NTP is to keep the time accurate based upon the time that the configured servers report. NTP uses UTC as its time standard, PTP uses TAI (International Atomic Time), there is a file (`MULTINET:LEAP-SECONDS.LIST`) that contains the leap seconds necessary to convert one to the other. NTP uses all configured clocks that respond with reasonable time and delay characteristics to determine what the correct time should be. PTP chooses a single master clock based upon the following reported information:

1. Priority One Field (`ptpengine:priority1`) This is a configuration file item and lowest wins. Normally set to 128 for a potential master and 255 for slave only.
2. Clock Class. This is based upon configured value (`ptpengine:clock_class`) and modified based upon actual operating state.
3. Clock Accuracy. This is based upon configuration (`ptpengine:ptp_clock_accuracy`) and computed accuracy scaled to an enumerated value.
4. Clock Variance. This is based upon configuration (`ptpengine:ptp_allan_variance`) and computed variance scaled to an enumerated value.
5. Priority 2 field (`ptpengine:priority2`) Configuration file item which can be used to identify primary and backup clocks among otherwise identical, redundant Grandmaster clocks.
6. Source Port ID. A unique value, generally the Ethernet MAC address.

PTP is not available on VAX systems or AXP V6.

## Enabling PTP in MultiNet

1. Enable the PTP server in MultiNet's server configuration utility:

```
$ MULTINET CONFIGURE/SERVER
MultiNet Server Configuration Utility V5.6
[Reading in configuration from MULTINET:SERVICES.MASTER_SERVER]
SERVER-CONFIG>ENABLE PTP
SERVER-CONFIG>EXIT
[Writing configuration to
MULTINET_COMMON_ROOT:[MULTINET]SERVICES.MASTER_SERVER]
```

```
$
```

2. Create `MULTINET:PTPD2.CONF` with information from the *Configuration* section of this chapter.

3. Create `MULTINET:ETHER.` with ethernet (MAC) addresses and names of possible masters. Each line has the format `ethernet-address nodename`

```
aa:00:04:00:01:01 ptp-master.example.com
```

4. Restart the MultiNet master server:

```
$ @MULTINET:START_SERVER RESTART
```

# Configuration

## Minimal Example

```
; =====  
; This is a minimal configuration for a PTPv2 slave  
; =====  
; interface has to be specified  
ptpengine:interface=se0  
; PTP domain  
ptpengine:domain=0  
; available presets are slaveonly, masteronly and masterslave (IEEE 1588)  
ptpengine:preset=slaveonly  
; multicast for both sync and delay requests - use hybrid for unicast delay  
; requests  
ptpengine:ip_mode=multicast  
; status file providing an overview of ptpd's operation and statistics  
global:log_status=y  
; required if ip_mode is set to hybrid  
;ptpengine:log_delayreq_interval=0  
; uncomment this to log a timing log  
;global:statistics_file=multinet:ptpd2.stats  
; always keep a new line at the end
```

## Available Configuration Options

**ptpengine:interface = *interface***

Network interface to use - se0, se1 etc. (required).

**ptpengine:backup\_interface = interface**

Backup network interface to use - se0, se1 etc. When no grandmaster is available, slave will keep alternating between primary and secondary until a grandmaster is found.

**ptpengine:preset = slaveonly**

PTP engine preset. Possible values are:

none	Defaults, no clock class restrictions
masteronly	Master, passive when not best master (clock class 0..127)
masterslave	Full IEEE 1588 implementation: Master, slave when not best master (clock class 128..254)
slaveonly	Slave only (clock class 255 only). Default setting.

**ptpengine:ip\_mode = multicast**

IP transmission mode. Possible values are:

multicast	uses multicast for all messages (the default)
hybrid	uses multicast for sync and announce, and unicast for delay request and response
unicast	uses unicast for all transmission.

When unicast mode is selected, destination IP(s) need to be configured using `ptpengine:unicast_destinations`.

**ptpengine:unicast\_negotiation = N**

Enable unicast negotiation support using signaling messages.

**ptpengine:disable\_bmca = N**

Disable Best Master Clock Algorithm for unicast masters: Only effective for `masteronly` preset - all Announce messages will be ignored and the clock will transition directly into MASTER state.

**ptpengine:unicast\_negotiation\_listening = N**

When unicast negotiation enabled on a master clock, reply to transmission requests also in LISTENING state.

**ptpengine:delay\_mechanism = E2E**

Delay detection mode used - use DELAY\_DISABLED for synchronization only (no full synchronization).

Options: E2E P2P DELAY\_DISABLED

**ptpengine:domain = 0**

PTP domain number.

**ptpengine:port\_number = 1**

PTP port number (part of PTP Port Identity - not UDP port). For ordinary clocks (single port), the default should be used, but when running multiple instances to simulate a boundary clock, the port number can be changed.

**ptpengine:any\_domain = N**

Usability extension: if enabled, a slave-only clock will accept masters from any domain, while preferring the configured domain, and preferring lower domain number.

NOTE: this behavior is not part of the standard.

**ptpengine:slave\_only = Y**

Slave only mode (sets clock class to 255, overriding value from preset).

**ptpengine:inbound\_latency = 0**

Specify latency correction (nanoseconds) for incoming packets.

**ptpengine:outbound\_latency = 0**

Specify latency correction (nanoseconds) for outgoing packets.

**ptpengine:offset\_shift = 0**

Apply an arbitrary shift (nanoseconds) to offset from master when in slave state. Value can be positive or negative - useful for correcting for antenna latencies, delay asymmetry and IP stack latencies. This will not be visible in the offset from master value - only in the resulting clock correction.

**ptpengine:always\_respect\_utc\_offset = N**

Compatibility option: In slave state, always respect UTC offset announced by best master, even if the `currentUtcOffsetValid` flag is announced `FALSE`.

NOTE: this behavior is not part of the standard.

**ptpengine:prefer\_utc\_offset\_valid = N**

Compatibility extension to BMC algorithm: when enabled, BMC for both master and slave clocks will prefer masters announcing `currentUtcOffsetValid` as `TRUE`.

NOTE: this behavior is not part of the standard.

**ptpengine:require\_utc\_offset\_valid = N**

Compatibility option: when enabled, `ptpd2` will ignore Announce messages from masters announcing `currentUtcOffsetValid` as `FALSE`.

NOTE: this behavior is not part of the standard.

**ptpengine:unicast\_grant\_duration = 300**

Time (seconds) unicast messages are requested for by slaves when using unicast negotiation, and maximum time unicast message transmission is granted to slaves by masters

**ptpengine:log\_announce\_interval = 1**

PTP announce message interval in master state. When using unicast negotiation, for slaves this is the minimum interval requested, and for masters this is the only interval granted. (expressed as log 2 i.e. -1=0.5s, 0=1s, 1=2s etc.)

**ptpengine:log\_announce\_interval\_max = 5**

Maximum Announce message interval requested by slaves when using unicast negotiation, (expressed as log 2 i.e. -1=0.5s, 0=1s, 1=2s etc.)

**ptpengine:announce\_receipt\_timeout = 6**

PTP announce receipt timeout announced in master state.

**ptpengine:announce\_receipt\_grace\_period = 0**

PTP announce receipt timeout grace period in slave state: when announce receipt timeout occurs, disqualify current best GM, then wait  $n$  times announce receipt timeout before resetting. Allows for a seamless GM failover when standby GMs are slow to react. When set to 0, this option is not used.

**ptpengine:log\_sync\_interval = 0**

PTP sync message interval in master state. When using unicast negotiation, for slaves this is the minimum interval requested, and for masters this is the only interval granted. (expressed as log 2 i.e. -1=0.5s, 0=1s, 1=2s etc.)

**ptpengine:log\_sync\_interval\_max = 5**

Maximum Sync message interval requested by slaves when using unicast negotiation, (expressed as log 2 i.e. -1=0.5s, 0=1s, 1=2s etc.)

**ptpengine:log\_delayreq\_override = N**

Override the Delay Request interval announced by best master.

**ptpengine:log\_delayreq\_auto = Y**

Automatically override the Delay Request interval if the announced value is 127 (0X7F), such as in unicast messages (unless using unicast negotiation).

**ptpengine:log\_delayreq\_interval\_initial = 0**

Delay request interval used before receiving first delay response (expressed as log 2 i.e. -1=0.5s, 0=1s, 1=2s etc.)

**ptpengine:log\_delayreq\_interval = 0**

Minimum delay request interval announced when in master state, in slave state overrides the master interval, required in hybrid mode. When using unicast negotiation, for slaves this is the minimum

interval requested, and for masters this is the minimum interval granted. (expressed as log 2 i.e. -1=0.5s, 0=1s, 1=2s etc.)

**ptpengine:log\_delayreq\_interval\_max = 5**

Maximum Delay Response interval requested by slaves when using unicast negotiation, (expressed as log 2 i.e. -1=0.5s, 0=1s, 1=2s etc.)

**ptpengine:log\_peer\_delayreq\_interval = 1**

Minimum peer delay request message interval in peer to peer delay mode. When using unicast negotiation, this is the minimum interval requested, and the only interval granted. (expressed as log 2 i.e. -1=0.5s, 0=1s, 1=2s etc.)

**ptpengine:log\_peer\_delayreq\_interval\_max = 5**

Maximum Peer Delay Response interval requested by slaves when using unicast negotiation (expressed as log 2 i.e. -1=0.5s, 0=1s, 1=2s etc.)

**ptpengine:foreignrecord\_capacity = 5**

Foreign master record size (Maximum number of foreign masters).

**ptpengine:ptp\_allan\_variance = 65535**

Specify Allan variance announced in master state.

**ptpengine:ptp\_clock\_accuracy = ACC\_UNKNOWN**

Clock accuracy range announced in master state.

Options: ACC\_25NS ACC\_100NS ACC\_250NS ACC\_1US ACC\_2.5US ACC\_10US  
ACC\_25US ACC\_100US ACC\_250US ACC\_1MS ACC\_2.5MS ACC\_10MS ACC\_25MS  
ACC\_100MS ACC\_250MS ACC\_1S ACC\_10S ACC\_10SPLUS ACC\_UNKNOWN

**ptpengine:utc\_offset = 0**

Underlying time source UTC offset announced in master state.



**ptpengine:utc\_offset\_valid = N**

Underlying time source UTC offset validity announced in master state.

**ptpengine:time\_traceable = N**

Underlying time source time traceability announced in master state.

**ptpengine:frequency\_traceable = N**

Underlying time source frequency traceability announced in master state.

**ptpengine:ptp\_timescale = PTP**

Time scale announced in master state (with ARB, UTC properties are ignored by slaves). When clock class is set to 13 (application specific), this value is ignored and ARB is used.

Options: PTP ARB

**ptpengine:ptp\_timesource = INTERNAL\_OSCILLATOR**

Time source announced in master state.

Options: ATOMIC\_CLOCK GPS TERRESTRIAL\_RADIO PTP NTP HAND\_SET OTHER  
INTERNAL\_OSCILLATOR

Note that case matters.

**ptpengine:clock\_class = 255**

Clock class - announced in master state. Always 255 for slave-only. Minimum, maximum and default values are controlled by presets. If set to 13 (application specific time source), announced time scale is always set to ARB. This setting controls the states a PTP port can be in. If below 128, port will only be in MASTER or PASSIVE states (master only). If above 127, port will be in MASTER or SLAVE states.

**ptpengine:priority1 = 128**

Priority 1 announced in master state, used for Best Master Clock selection.

**ptpengine:priority2 = 128**

Priority 2 announced in master state, used for Best Master Clock selection.

**ptpengine:max\_listen = 5**

Number of consecutive resets to LISTENING before full network reset

**ptpengine:unicast\_destinations =**

Specify unicast slave addresses for unicast master operation, or unicast master addresses for slave operation. Format is: comma, tab or space-separated IPv4 unicast addresses, one or more. For a slave, when unicast negotiation is used, setting this is mandatory.

**ptpengine:unicast\_domains =**

This is only used by slave-only clocks using unicast destinations to allow for each master to be in a separate domain, such as with Telecom Profile. The number of entries should match the number of unicast destinations, otherwise unconfigured domains or domains set to 0 are set to domain configured in `ptpengine:domain`. The format is a comma, tab, or space-separated list of 8-bit unsigned integers (0 ... 255)

**ptpengine:unicast\_local\_preference =**

Specify a local preference for each configured unicast destination

(`ptpengine:unicast_destinations`). This is only used by slave-only clocks using unicast destinations to allow for each master's BMC selection to be influenced by the slave, such as with Telecom Profile. The number of entries should match the number of unicast destinations, otherwise unconfigured preference is set to 0 (highest). The format is a comma, tab or space-separated list of 8-bit unsigned integers (0 ... 255).

**ptpengine:unicast\_peer\_destination =**

Specify peer unicast address for P2P unicast. Mandatory when running unicast mode and P2P delay mode.

**ptpengine:management\_set\_enable = N**

Enable handling of PTP management messages.

**ptpengine:management\_enable = Y**

Accept SET and COMMAND management messages.

**ptpengine:igmp\_refresh = Y**

Send explicit IGMP joins between engine resets and periodically in master state.

**ptpengine:master\_igmp\_refresh\_interval = 60**

Periodic IGMP join interval (seconds) in master state when running IPv4 multicast: when set below 10 or when ptpengine:igmp\_refresh is disabled, this setting has no effect.

**ptpengine:multicast\_ttl = 64**

Multicast time to live for multicast PTP packets (ignored and set to 1 for peer to peer messages).

**ptpengine:ip\_dscp = 0**

DiffServ CodePoint for packet prioritization (decimal). When set to zero, this option is not used. Use 46 for Expedited Forwarding (0x2e).

**ptpengine:sync\_stat\_filter\_enable = N**

Enable statistical filter for Sync messages.

**ptpengine:sync\_stat\_filter\_type = min**

Type of filter used for Sync message filtering. Options: none mean min max absmin absmax median

**ptpengine:sync\_stat\_filter\_window = 4**

Number of samples used for the Sync statistical filter

**ptpengine:sync\_stat\_filter\_window\_type = sliding**

Sample window type used for Sync message statistical filter. Delay Response outlier filter action.

Sliding window is continuous, interval passes every n-th sample only. Options: sliding interval

**ptpengine:delay\_stat\_filter\_enable = N**

Enable statistical filter for Delay messages.

**ptpengine:delay\_stat\_filter\_type = min**

Type of filter used for Delay message statistical filter. Options: none mean min max absmin  
absmax median

**ptpengine:delay\_stat\_filter\_window = 4**

Number of samples used for the Delay statistical filter

**ptpengine:delay\_stat\_filter\_window\_type = sliding**

Sample window type used for Delay message statistical filter. Sliding window is continuous, interval  
passes every n-th sample only. Options: sliding interval

**ptpengine:delay\_outlier\_filter\_enable = N**

Enable outlier filter for the Delay Response component in slave state.

**ptpengine:delay\_outlier\_filter\_action = discard**

Delay Response outlier filter action. If set to 'filter', outliers are replaced with moving average.

Options: discard filter

**ptpengine:delay\_outlier\_filter\_capacity = 20**

Number of samples in the Delay Response outlier filter buffer.

**ptpengine:delay\_outlier\_filter\_threshold = 1.000000**

Delay Response outlier filter threshold (multiplier for Peirce's maximum standard deviation). When set  
below 1.0, filter is tighter, when set above 1.0, filter is looser than standard Peirce's test. When autotune  
is enabled, this is the starting threshold.

**ptpengine:delay\_outlier\_filter\_always\_filter = N**

Always run the Delay Response outlier filter, even if clock is being slewed at maximum rate

**ptpengine:delay\_outlier\_filter\_autotune\_enable = Y**

Enable automatic threshold control for Delay Response outlier filter.

**ptpengine:delay\_outlier\_filter\_autotune\_minpercent = 20**

Delay Response outlier filter autotune low watermark - minimum percentage of discarded samples in the update period before filter is tightened by the autotune step value.

**ptpengine:delay\_outlier\_filter\_autotune\_maxpercent = 95**

Delay Response outlier filter autotune high watermark - maximum percentage of discarded samples in the update period before filter is loosened by the autotune step value.

**ptpengine:delay\_outlier\_autotune\_step = 0.100000**

The value the Delay Response outlier filter threshold is increased or decreased by when auto-tuning.

**ptpengine:delay\_outlier\_filter\_autotune\_minthreshold = 0.100000**

Minimum Delay Response filter threshold value used when auto-tuning.

**ptpengine:delay\_outlier\_filter\_autotune\_maxthreshold = 5.000000**

Maximum Delay Response filter threshold value used when auto-tuning.

**ptpengine:delay\_outlier\_filter\_stepdetect\_enable = N**

Enable Delay filter step detection (delaySM) to block when certain level exceeded.

**ptpengine:delay\_outlier\_filter\_stepdetect\_threshold = 1000000**

Delay Response step detection threshold. Step detection is performed only when delaySM is below this threshold (nanoseconds).

**ptpengine:delay\_outlier\_filter\_stepdetect\_level = 500000**

Delay Response step level. When step detection enabled and operational, delaySM above this level (nanosecond) is considered a clock step and updates are paused.

**ptpengine:delay\_outlier\_filter\_stepdetect\_credit = 200**

Initial credit (number of samples) the Delay step detection filter can block for. When credit is exhausted, filter stops blocking. Credit is gradually restored.

**ptpengine:delay\_outlier\_filter\_stepdetect\_credit\_increment = 10**

Amount of credit for the Delay step detection filter restored every full sample window

**ptpengine:delay\_outlier\_weight = 1.000000**

Delay Response outlier weight: if an outlier is detected, determines the amount of its deviation from mean that is used to build the standard deviation statistics and influence further outlier detection. When set to 1.0, the outlier is used as is.

**ptpengine:sync\_outlier\_filter\_enable = N**

Enable outlier filter for the Sync component in slave state.

**ptpengine:sync\_outlier\_filter\_action = discard**

Sync outlier filter action. If set to 'filter', outliers are replaced with moving average. Options: discard filter

**ptpengine:sync\_outlier\_filter\_capacity = 20**

Number of samples in the Sync outlier filter buffer.

**ptpengine:sync\_outlier\_filter\_threshold = 1.000000**

Sync outlier filter threshold: multiplier for the Peirce's maximum standard deviation. When set below 1.0, filter is tighter, when set above 1.0, filter is looser than standard Peirce's test.

**ptpengine:sync\_outlier\_filter\_always\_filter = N**

Always run the Sync outlier filter, even if clock is being slewed at maximum rate

**ptpengine:sync\_outlier\_filter\_autotune\_enable = Y**

Enable automatic threshold control for Sync outlier filter.

**ptpengine:sync\_outlier\_filter\_autotune\_minpercent = 20**

Sync outlier filter autotune low watermark - minimum percentage of discarded samples in the update period before filter is tightened by the autotune step value.

**ptpengine:sync\_outlier\_filter\_autotune\_maxpercent = 95**

Sync outlier filter autotune high watermark - maximum percentage of discarded samples in the update period before filter is loosened by the autotune step value.

**ptpengine:sync\_outlier\_autotune\_step = 0.100000**

Value the Sync outlier filter threshold is increased or decreased by when auto-tuning.

**ptpengine:sync\_outlier\_filter\_autotune\_minthreshold = 0.100000**

Minimum Sync outlier filter threshold value used when auto-tuning

**ptpengine:sync\_outlier\_filter\_autotune\_maxthreshold = 5.000000**

Maximum Sync outlier filter threshold value used when auto-tuning

**ptpengine:sync\_outlier\_filter\_stepdetect\_enable = N**

Enable Sync filter step detection (delayMS) to block when certain level exceeded.

**ptpengine:sync\_outlier\_filter\_stepdetect\_threshold = 1000000**

Sync step detection threshold. Step detection is performed only when delayMS is below this threshold (nanoseconds)

**ptpengine:sync\_outlier\_filter\_stepdetect\_level = 500000**

Sync step level. When step detection enabled and operational, delayMS above this level (nanosecond) is considered a clock step and updates are paused

**ptpengine:sync\_outlier\_filter\_stepdetect\_credit = 200**

Initial credit (number of samples) the Sync step detection filter can block for. When credit is exhausted, filter stops blocking. Credit is gradually restored

**ptpengine:sync\_outlier\_filter\_stepdetect\_credit\_increment = 10**

Amount of credit for the Sync step detection filter restored every full sample window

**ptpengine:sync\_outlier\_weight = 1.000000**

Sync outlier weight: if an outlier is detected, this value determines the amount of its deviation from mean that is used to build the standard deviation statistics and influence further outlier detection. When set to 1.0, the outlier is used as is.

**ptpengine:calibration\_delay = 0**

Delay between moving to slave state and enabling clock updates (seconds). This allows one-way delay to stabilize before starting clock updates. Activated when going into slave state and during slave's GM failover.

**ptpengine:idle\_timeout = 120**

PTP idle timeout: if PTPd is in SLAVE state and there have been no clock updates for this amount of time, PTPd releases clock control. Measured in seconds.

**ptpengine:panic\_mode = N**

Enable panic mode: when offset from master is above 1 second, stop updating the clock for a period of time and then step the clock if offset remains above 1 second.

**ptpengine:panic\_mode\_duration = 2**

Duration (minutes) of the panic mode period (no clock updates) when offset above 1 second detected.

**ptpengine:panic\_mode\_release\_clock = N**

When entering panic mode, release clock control while panic mode lasts. If `ntpengine:*` configured, this will fail over to NTP, if not set, PTP will hold clock control during panic mode.



**ptpengine:panic\_mode\_exit\_threshold = 0**

Do not exit panic mode until offset drops below this value (nanoseconds).

0 = not used.

**ptpengine:pid\_as\_clock\_identity = N**

Use PTP's process ID as the middle part of the PTP clock ID - useful for running multiple instances.

**ptpengine:ntp\_failover = N**

Fail over to NTP when PTP time sync not available – requires `ptpengine:enabled`, but does not require the rest of NTP configuration: will warn instead of failing over if cannot control `ntpd`.

**ptpengine:ntp\_failover\_timeout = 120**

NTP failover timeout in seconds: time between PTP slave going into LISTENING state and releasing clock control.

0 = fail over immediately.

**ptpengine:prefer\_ntp = N**

Prefer NTP time synchronization. Only use PTP when NTP not available, could be used when NTP runs with a local GPS receiver or another reference

**ptpengine:panic\_mode\_ntp = N**

Same as `ptpengine:panic_mode_release_clock`

**ptpengine:timing\_acl\_permit =**

Permit access control list for timing packets. Format is a series of comma, space or tab separated network prefixes: IPv4 addresses or full CIDR notation `a.b.c.d/x`, where `a.b.c.d` is the subnet and `x` is the decimal mask, or `a.b.c.d/v.x.y.z` where `a.b.c.d` is the subnet and `v.x.y.z` is the 4-octet mask. The match is performed on the source IP address of the incoming messages. No addresses are allowed unless an ACL is defined.

**ptpengine:timing\_acl\_deny =**

Deny access control list for timing packets. Format is a series of comma, space or tab separated network prefixes: IPv4 addresses or full CIDR notation a.b.c.d/x, where a.b.c.d is the subnet and x is the decimal mask, or a.b.c.d/v.x.y.z where a.b.c.d is the subnet and v.x.y.z is the 4-octet mask. The match is performed on the source IP address of the incoming messages. No addresses are allowed unless an acl is defined.

**ptpengine:management\_acl\_permit =**

Permit access control list for management messages. Format is a series of comma, space or tab separated network prefixes: IPv4 addresses or full CIDR notation a.b.c.d/x, where a.b.c.d is the subnet and x is the decimal mask, or a.b.c.d/v.x.y.z where a.b.c.d is the subnet and v.x.y.z is the 4-octet mask. The match is performed on the source IP address of the incoming messages. No addresses are allowed unless an acl is defined.

**ptpengine:management\_acl\_deny =**

Deny access control list for management messages. Format is a series of comma, space or tab separated network prefixes: IPv4 addresses or full CIDR notation a.b.c.d/x, where a.b.c.d is the subnet and x is the decimal mask, or a.b.c.d/v.x.y.z where a.b.c.d is the subnet and v.x.y.z is the 4-octet mask. The match is performed on the source IP address of the incoming messages. No addresses are allowed unless an acl is defined.

**ptpengine:timing\_acl\_order = permit-deny**

Order in which permit and deny access lists are evaluated for timing packets. The IP address that the packet came from is checked for a match in the permit list and deny list. When the order is permit-deny (default) the address is only permitted if it is in the permit list and is not in the deny list. For deny-permit the address is allowed if it is not in the deny list or if it is in the permit list, otherwise it is rejected.

Options: permit-deny deny-permit

**ptpengine:management\_acl\_order = permit-deny**

Order in which permit and deny access lists are evaluated for management messages. The IP address that the packet came from is checked for a match in the permit list and deny list. When the order is permit-deny (default) the address is only permitted if it is in the permit list and is not in the deny list. For deny-permit the address is allowed if it is not in the deny list or if it is in the permit list, otherwise it is rejected.

Options: permit-deny deny-permit

**ptpengine:sync\_sequence\_checking = N**

When enabled, Sync messages will only be accepted if sequence ID is increasing. This is limited to 50 dropped messages.

**ptpengine:clock\_update\_timeout = 0**

If set to non-zero, timeout in seconds, after which the slave resets if no clock updates made.

**clock:no\_adjust = N**

Do not adjust the clock.

**clock:no\_reset = N**

Do not step the clock - only slew.

**clock:step\_startup\_force = N**

Force clock step on first sync after startup regardless of offset and `clock:no_reset`

**clock:step\_startup = N**

Step clock on startup if offset  $\geq$  1 second, ignoring panic mode and `clock:no_reset`

**clock:drift\_handling = preserve**

Observed drift handling method between servo restarts:

reset: set to zero (not recommended)

preserve: use kernel value,

file: load/save to drift file on startup/shutdown, use kernel value in between. To specify drift file, use the `clock:drift_file` setting.

Options: reset preserve file

**clock:drift\_file = MULTINET:ptpd2\_kernelclock.drift**

Specify drift file.

**clock:leap\_second\_pause\_period = 5**

Time (seconds) before and after midnight that clock updates should be suspended for during a leap second event. The total duration of the pause is twice the configured duration.

**clock:leap\_second\_notice\_period = 43200**

Time (seconds) before midnight that PTPd starts announcing the leap second if it's running as master

**clock:leap\_seconds\_file =**

Specify leap second file location - up to date version can be downloaded from:

<http://www.ietf.org/timezones/data/leap-seconds.list>

**clock:leap\_second\_handling = accept**

Behavior during a leap second event:

accept: inform the OS kernel of the event

ignore: do nothing - ends up with a 1-second offset which is then slewed

step: similar to ignore, but steps the clock immediately after the leap second event

smear: do not inform kernel, gradually introduce the leap second before the event by modifying clock offset (see `clock:leap_second_smear_period`)

Options: accept ignore step smear

**clock:leap\_second\_smear\_period = 86400**

Time period (Seconds) over which the leap second is introduced before the event.

Example: when set to 86400 (24 hours), an extra 11.5 microseconds is added every second

**clock:max\_offset\_ppm = 500**

Maximum absolute frequency shift which can be applied to the clock servo when slewing the clock. Expressed in parts per million (1 ppm = shift of 1 ms per second). Values above 512 will use the tick duration correction to allow even faster slewing. Default maximum is 512 without using tick.

**servo:dt\_method = constant**

How servo update interval (delta t) is calculated:

none: servo not corrected for update interval (dt always 1),

constant: constant value (target servo update rate - sync interval for PTP,

measured: servo measures how often it's updated and uses this interval.

Options: none constant measured

**servo:delayfilter\_stiffness = 6**

One-way delay filter stiffness.

**servo:kp = 0.100000**

Clock servo PI controller proportional component gain (kP).

**servo:ki = 0.001000**

Clock servo PI controller integral component gain (kI).

**servo:dt\_max = 5.000000**

Maximum servo update interval (delta t) when using measured servo update interval (servo:dt\_method = measured), specified as sync interval multiplier.

**servo:stability\_detection = N**

Enable clock synchronization servo stability detection (based on standard deviation of the observed drift value) - drift will be saved to drift file / cached when considered stable, also clock stability status will be logged.

**servo:stability\_threshold = 10.000000**

Specify the observed drift standard deviation threshold in parts per billion (ppb) - if standard deviation is within the threshold, servo is considered stable.

**servo:stability\_period = 1**

Specify for how many statistics update intervals the observed drift standard deviation has to stay within threshold to be considered stable.

**servo:stability\_timeout = 10**

Specify after how many minutes without stabilization servo is considered unstable. Assists with logging servo stability information and allows to preserve observed drift if servo cannot stabilize.

**servo:max\_delay = 0**

Do not accept master to slave delay (delayMS - from Sync message) or slave to master delay (delaySM - from Delay messages) if greater than this value (nanoseconds). 0 = not used.

**servo:max\_delay\_max\_rejected = 0**

Maximum number of consecutive delay measurements exceeding maxDelay threshold, before slave is reset.

**servo:max\_delay\_stable\_only = N**

If `servo:max_delay` is set, perform the check only if clock servo has stabilized.

**servo:max\_offset = 0**

Do not reset the clock if offset from master is greater than this value (nanoseconds). 0 = not used.

**global:lock\_file =MULTINET:PTPD2.LOCK**

Lock file location.

**global:auto\_lockfile = N**

Use mode specific and interface specific lock file (overrides `global:lock_file`).

**global:lock\_directory = MULTINET:**

Lock file directory: used with automatic mode-specific lock files, also used when no lock file is specified. When lock file is specified, it's expected to be an absolute path.

**global:ignore\_lock = N**

Skip lock file checking and locking.

**global:quality\_file =**

File used to record data about sync packets. Enables recording when set.

**global:quality\_file\_max\_size = 0**

Maximum sync packet record file size (in kB) - file will be truncated if size exceeds the limit. 0 - no limit.

**global:quality\_file\_max\_files = 0**

Enable log rotation of the sync packet record file up to n files.

0 - do not rotate.

**global:quality\_file\_truncate = N**

Truncate the sync packet record file every time it is (re) opened (on startup or restart).

**global:status\_file = MULTINET:ptpd2.status**

File used to log ptpd2 status information.

**global:log\_status = N**

Enable / disable writing status information to file.

**global:status\_update\_interval = 1**

Status file update interval in seconds.

**global:log\_level = LOG\_ALL**

Specify log level (only messages at this priority or higher will be logged). The minimal level is LOG\_ERR.

Options: LOG\_ERR LOG\_WARNING LOG\_NOTICE LOG\_INFO LOG\_ALL

**global:statistics\_file =**

Specify statistics log file path. Setting this enables logging of statistics but can be overridden with global:log\_statistics.

**global:statistics\_log\_interval = 0**

Log timing statistics every *n* seconds for Sync and Delay messages

(0 - log all).

**global:statistics\_file\_max\_size = 0**

Maximum statistics log file size (in kB) - log file will be truncated if size exceeds the limit. 0 - no limit.

**global:statistics\_file\_max\_files = 0**

Enable log rotation of the statistics file up to *n* files.

0 - do not rotate.

**global:statistics\_file\_truncate = N**

Truncate the statistics file every time it is (re) opened (startup and restart).

**global:dump\_packets = N**

Dump the contents of every PTP packet

**global:log\_statistics = N**

Log timing statistics for every PTP packet received



**global:statistics\_timestamp\_format = datetime**

Timestamp format used when logging timing statistics

(when global:log\_statistics is enabled):

`datetime` - formatted date and time: YYYY-MM-DD hh:mm:ss.uuuuuu

`unix` - Unix timestamp with nanoseconds: s.ns

`both` - Formatted date and time, followed by UNIX timestamp (adds one extra field to the log)

Options: `datetime` `unix` `both`

**global:statistics\_update\_interval = 30**

Clock synchronization statistics update interval in seconds.

**global:periodic\_updates = N**

Log a status update every time statistics are updated (`global:statistics_update_interval`). The updates are logged even when `ptpd` is configured without statistics support.

**global:timingdomain\_election\_delay = 15**

Delay (seconds) before releasing a time service (NTP or PTP) and electing a new one to control a clock. 0 = elect immediately

**global:enable\_snmp=N/Y**

Use the AgentX protocol to connect to the SNMP agent on the system to provide information about the server.

**ntpengine:enabled = N**

Enable NTPd integration.

**ntpengine:control\_enabled = N**

Enable control over local NTPd daemon.

**ntpengine:check\_interval = 15**

NTP control check interval in seconds.

**ntpengine:key\_id = 0**

NTP key number - must be configured as a trusted control key in `ntp.conf`, and be non-zero for the `ntpengine:control_enabled` setting to take effect.

**ntpengine:key =**

NTP key (plain text, max. 20 characters) - must match the key configured in ntpd's keys file, and must be non-zero for the `ntpengine:control_enabled` setting to take effect.

# MultiNet NetControl PTP Commands

The following commands are available in MultiNet's NETCONTROL utility to manage the PTP server.

## DEBUG

Set debugging level

```
$ mult netc ptp debug 0  
Connected to NETCONTROL server on "LOCALHOST"  
bigboote.example.com Network Control V5.6 at Fri 16-Aug-2019 1:36PM-EDT  
Debug level now set to 0
```

## HELP

List of PTP control commands

```
$ mult netc ptp help  
Connected to NETCONTROL server on "LOCALHOST"  
bigboote.example.com Network Control V5.6 at Fri 16-Aug-2019 1:41PM-EDT  
debug - set debugging level  
help - this help information  
noop - no operation  
ptp-control-version - version of netcontrol control  
reload - restart PTP  
restart - restart PTP  
show - show operating information about PTP  
shutdown - shutdown PTP  
start - start PTP  
version - version of PTP
```

## RELOAD

Restart/reload PTP

```
$ mult netc ptp reload  
Connected to NETCONTROL server on "LOCALHOST"  
bigboote.example.com Network Control V5.6 at Fri 16-Aug-2019 1:37PM-EDT  
PTP server restarting  
PTP server restart requested
```

# SHOW

Show operating information about PTP

```
$ mult netc ptp show
Connected to NETCONTROL server on "LOCALHOST"
bigboote.example.com Network Control V5.6 at Fri 16-Aug-2019 1:40PM-EDT
Offset From Master 0 0
Mean Path Delay 0 0
observed parent clock phase change rate 0
Grand Master Identity aa 0 4 ff fe 0 ae 8 (sys1.example.com)
Grand Master clock quality 248 254 65535
Steps Removed 0
Clock Class 248
Clock Accuracy 254
Offset Scaled Log Variance 0
Domain Number 0
End of Show PTP
```

# SHUTDOWN

Shutdown PTP

```
$ mult netc ptp shutdown
Connected to NETCONTROL server on "LOCALHOST"
sys1.example.com Network Control V5.6 at Fri 16-Aug-2019 1:40PM-EDT
Starting shutdown of PTP server
PTP server shutdown
```

# START

Start PTP

```
$ mult netc ptp shutdown
Connected to NETCONTROL server on "LOCALHOST"
bigboote.example.com Network Control V5.6 at Fri 16-Aug-2019 1:40PM-EDT
Starting shutdown of PTP server
PTP server shutdown
```

# VERSION

Version of MultiNet PTP and PTP

```
$ mult netc ptp version
```

```
Connected to NETCONTROL server on "LOCALHOST"
```

```
bigboote.example.com Network Control V5.6 at Fri 16-Aug-2019 1:36PM-EDT
```

```
PTP for MultiNet V1.0
```

```
PTP server version = 2.0(1)
```

# Files

## **MULTINET: PTPD2.CONF**

System specific configuration

## **MULTINET: ETHER.**

Optional file containing a list of ethernet mac addresses and host names for displaying. One line per host name in the format

```
XX:XX:XX:XX:XX:XX name_to_be_displayed
```

## **MULTINET: LEAP-SECONDS.LIST**

List of dates in which a leap second has been added to adjust from Coordinated Universal Time (UTC) to International Atomic Time (TAI).

# Appendix A. Server Configuration Parameters

## SERVER-CONFIG Service Parameters

The below table describes the service parameters you can set with the SERVER-CONFIG Utility.

Parameter	Description
ACCEPT-HOSTS	The list of hosts allowed access to this server. See Chapter 10.
ACCEPT-NETS	The list of networks or subnetworks that are allowed to access this server. See Chapter 10.
BACKLOG	The number of server connections to queue up before refusing to accept additional connections when MAX-SERVERS is reached.
CONFFILE	This file is used instead of the default MULTINET:NAMED.CONF file.
CONNECTED	The name of the internal MULTINET_SERVER routine to call when a connection request is received. This varies by protocol and is normally not changed. For servers supplied with MultiNet, do not change this parameter. When adding your own servers, you usually do not need to modify the default parameters.
DEBUG	Sets the debug level of the Domain Nameserver (the default is no debugging). The larger the number, the more verbose the output. A value of 0 turns off debugging.
DEBUGFILE	This file is used instead of the default MULTINET:NAMED.RUN file.
DISABLED-NODES	The list of VMScluster nodes that cannot run this server. See Chapter 10.

ENABLED-NODES	The list of VMScluster nodes that can run this server. See Chapter 10.
FLAGS	Various flags that control the operation of the service.
INIT	The name of the internal MULTINET_SERVER routine for initializing a service. This varies by protocol and is normally not changed.
LISTEN	The name of the internal MULTINET_SERVER routine for listening for connections to the service. This varies by protocol and is normally not changed.
LOG-ACCEPTS	Specifies whether to log successful connections to the service.
LOG-FILE	Destination of log messages. May be a VMS filename or OPCOM to direct messages to the VMS OPCOM process.
LOG-REJECTS	Specifies whether to log rejected connections to the service. A connection is rejected because of the combination of the REJECT-HOSTS, REJECT-NETS, and REJECT-BY-DEFAULT parameters.
MAX-SERVERS	The maximum number of service processes to allow at any one time. If this limit is reached, additional connections - up to the number specified by the BACKLOG parameter - are accepted but not processed until one of the previous connections completes.
MAXIMUM-TTL	Changes the maximum time-to-live (TTL) that resource records are cached from the default of 604800 seconds (1 week) to the specified value.
MINIMUM-TTL	Changes the minimum time-to-live (TTL) that resource records are cached from the default of zero (0) seconds to the specified value.  It is recommended you use this command only if there is a specific need. This could cause problems in that you may be caching resource records for longer than the authoritative administrator intended.

PARAMETERS	Specifies service-dependent parameters passed to the initialization routine of built-in services. Normally not used for user-written services.
PRESERVE-CASE	Tells NAMED to keep the case of the response from the remote nameserver after doing a query.
PRIORITY	The VMS process priority to assign to created processes.
PROGRAM	The VMS filename of the image to run or merge.
QUERY-LOG	Toggles query logging ON and OFF. Query logging shows an informational message every time a query is received by the server. Query logging can be directed to OPCOM or a file in the MULTINET:NAMED.CONF file using the logging category <code>queries</code> .
REJECT-BY-DEFAULT	Whether to reject a connection from a host that does not match any of the ACCEPT-HOSTS, ACCEPT-NETS, REJECT-HOSTS, and REJECT-NETS lists.
REJECT-HOSTS	The list of hosts not allowed access to this server. See Chapter 10.
REJECT-MESSAGE	A text string to send down the network connection when a service is rejected.
REJECT-NETS	The list of networks or subnetworks not allowed access to this server. See Chapter 10.
REWRITE-TTL	Sets the time-to-live (TTL) that load balanced resource records are cached from the default of 300 seconds (5 minutes) to the specified value.
SERVICE	The name of the internal MULTINET_SERVER routine to call to perform the service. This is normally <code>Run_Program</code> for user-written services.
SERVICE-NAME	The name of the service.
SOCKET-FAMILY	The address family of the service; for example, <code>AF_INET</code> .



SOCKET-OPTIONS	Socket options to be set via <code>setsockopt()</code> . See the <i>MultiNet Programmer's Reference</i> for more information on socket options.
SOCKET-PORT	The port number on which to listen for connections.
SOCKET-TYPE	The type of socket; for example, <code>SOCK_STREAM</code> (TCP) or <code>SOCK_DGRAM</code> (UDP).
WORKING-SET	The VMS working set to assign to created processes.

## Services Provided with MultiNet

This table shows the MultiNet servers that can be disabled or enabled using `SERVER-CONFIG`.

Protocol	Server	Service Provided
RPC	BWNFS	Beame & Whiteside PC-NFS daemon
	NFS	Network File System server
	PCNFS	PC-NFS daemon
	RPCBOOTPARAM	RPC boot parameters for diskless hosts
	RPCLOCKMGR	RPC lock manager
	RPCMOUNT	RPC procedure for mounting file systems
	RPCPORTMAP	RPC portmapper (RPC naming service)
	RPCQUOTAD	Returns disk quota information
	RPCSTATUS	RPC status daemon

	RUSERS	Returns information about logged in users
	RWALL	Broadcasts messages to users
Special	CLUSTERALIAS	Special server for managing cluster-wide IP addresses
	RARP	Ethernet Reverse Address Resolution Protocol
	UCXQIO	VMS/ULTRIX Connection \$QIO emulation special services
	VIADECNET	Special server for IP-over-DECnet links
TCP	CHARGEN	Character generator
	DAYTIME	Returns time of day in ASCII
	DISCARD	Discard data received
	ECHO	Echo data received
	FINGER	Lists information about users on host
	FTP	File Transfer Protocol
	LPD	Remote printing service
	NETCONTROL	Remote network server control
	NETSTAT	Return network configuration and statistics
	NTALK	Interactive conversation with remote user (newer 4.3 BSD protocol)
	POP2	Post Office Protocol Version 2
	POP3	Post Office Protocol Version 3

	REXEC	Remote command execution (with password)
	RLOGIN	Network virtual terminal service (with BSD "R" services authentication)
	RSHELL	Remote command execution (with BSD "R" services authentication)
	SMTP	Simple Mail Transfer Protocol
	SSH	Secure Shell server and client
	SYSTAT	Remote system status (remote SHOW SYSTEM)
	TELNET	Network virtual terminal service
	TIME	Returns time of day in binary
<b>UDP</b>	BOOTP	Remote booting protocol
	DHCP	Dynamic remote booting protocol
	DOMAINNAME	Internet Domain Name System (DNS) name service (BIND)
	GATED	Gateway routing service (EGP, RIP, and HELLO)
	NTP	Network Time Protocol-For time synchronization
	SNMP	Simple Network Management Protocol agent
	SMUX	SNMP Multiplexing Protocol
	TALK	Interactive conversation with remote user (old 4.2 BSD protocol)
	TFTP	Trivial File Transfer Protocol
	UDPCHARGEN	Character generator

	UDPDAYTIME	Returns current time of day in ASCII
	UDPDISCARD	Discard data received
	UDPECHO	Echo data received
	UDPTIME	Returns time of day in binary

## Default Server Values

The below table shows the default value for MultiNet service parameters set using `SERVER-CONFIG`. Note that not all service parameters have defaults; hence, only those with default values are listed in this table.

**Note:** Many default values in this table will be shown as `Current process JPI$_xxxxx`. These will generally, but not always, correspond to the OpenVMS `SYSGEN` parameter `PQL_Dxxxxx`. For example, `JPI$_ASTLM` will generally correspond to `PQL_DASTLM`.

Server	Parameter	Default value
- All Services -	BACKLOG	10
	LOG-ACCEPTS	
	LOG-FILE	no
	LOG-REJECTS	
	MAX-SERVERS	none
	PRIORITY	
	REJECT-MESSAGE	no
	USERNAME	
	WORKING-SET-EXTENT	1000
	WORKING-SET-QUOTA	
		Current process JPI\$_PRIB
		none

		no Current process JPI\$_WSEXTENT Current process JPI\$_WSQUOTA
ACCOUNTING	PQL-ASTLM PQL-BIOLM PQL-BYTLM PQL-CPULM PQL-DIOLM PQL-ENQLM PQL-FILLM PQL-JTQUOTA PQL-PGFLQUOTA PQL-PRCLM PQL-TQELM	250 150 100000 0 150 500 100 4096 65536 0 100
BOOTP	PQL-ASTLM PQL-BIOLM PQL-BYTLM PQL-CPULM PQL-DIOLM PQL-ENQLM PQL-FILLM PQL-JTQUOTA PQL-PGFLQUOTA PQL-PRCLM PQL-TQELM	Current process JPI\$_ASTLM Current process JPI\$_BIOLM Current process JPI\$_BYTLM Current process JPI\$_CPULM Current process JPI\$_DIOLM Current process JPI\$_ENQLM Current process JPI\$_FILLM Current process JPI\$_JTQUOTA Current process JPI\$_PGFLQUOTA Current process JPI\$_PRCLM

		Current process JPI\$_TQELM
BWNFSD	PQL-ASTLM PQL-BIOLM PQL-BYTLM PQL-CPULM PQL-DIOLM PQL-ENQLM PQL-FILLM PQL-JTQUOTA PQL-PGFLQUOTA PQL-PRCLM PQL-TQELM	Current process JPI\$_ASTLM Current process JPI\$_BIOLM Current process JPI\$_BYTLM Current process JPI\$_CPULM Current process JPI\$_DIOLM Current process JPI\$_ENQLM Current process JPI\$_FILLM Current process JPI\$_JTQUOTA Current process JPI\$_PGFLQUOTA Current process JPI\$_PRCLM Current process JPI\$_TQELM
CHARGEN	PQL-ASTLM PQL-BIOLM PQL-BYTLM PQL-CPULM PQL-DIOLM PQL-ENQLM PQL-FILLM PQL-JTQUOTA PQL-PGFLQUOTA PQL-PRCLM PQL-TQELM	Current process JPI\$_ASTLM Current process JPI\$_BIOLM Current process JPI\$_BYTLM Current process JPI\$_CPULM Current process JPI\$_DIOLM Current process JPI\$_ENQLM Current process JPI\$_FILLM Current process JPI\$_JTQUOTA Current proces JPI\$_PGFLQUOTA Current process JPI\$_PRCLM Current process JPI\$_TQELM

CLUSTERALIAS	PQL-ASTLM PQL-BIOLM PQL-BYTLM PQL-CPULM PQL-DIOLM PQL-ENQLM PQL-FILLM PQL-JTQUOTA PQL-PGFLQUOTA PQL-PRCLM PQL-TQELM	Current process JPI\$_ASTLM  Current process JPI\$_BIOLM  Current process JPI\$_BYTLM  Current process JPI\$_CPULM  Current process JPI\$_DIOLM  Current process JPI\$_ENQLM  Current process JPI\$_FILLM  Current process JPI\$_JTQUOTA  Current proces JPI\$_PGFLQUOTA  Current process JPI\$_PRCLM  Current process JPI\$_TQELM
DAYTIME	PQL-ASTLM PQL-BIOLM PQL-BYTLM PQL-CPULM PQL-DIOLM PQL-ENQLM PQL-FILLM PQL-JTQUOTA PQL-PGFLQUOTA PQL-PRCLM PQL-TQELM	Current process JPI\$_ASTLM  Current process JPI\$_BIOLM  Current process JPI\$_BYTLM  Current process JPI\$_CPULM  Current process JPI\$_DIOLM  Current process JPI\$_ENQLM  Current process JPI\$_FILLM  Current process JPI\$_JTQUOTA  Current proces JPI\$_PGFLQUOTA  Current process JPI\$_PRCLM  Current process JPI\$_TQELM
DHCLIENT	PQL-ASTLM PQL-BIOLM PQL-BYTLM PQL-CPULM	250  150

	PQL-DIOLM PQL-ENQLM PQL-FILLM PQL-JTQUOTA PQL-PGFLQUOTA PQL-PRCLM PQL-TQELM	100000  0  150  500  100  4096  65536  0  100
DHCP	PQL-ASTLM PQL-BIOLM PQL-BYTLM PQL-CPULM PQL-DIOLM PQL-ENQLM PQL-FILLM PQL-JTQUOTA PQL-PGFLQUOTA PQL-PRCLM PQL-TQELM	250  150  100000  0  150  500  100  4096  65536  0  100
DISCARD	PQL-ASTLM PQL-BIOLM PQL-BYTLM PQL-CPULM PQL-DIOLM PQL-ENQLM PQL-FILLM PQL-JTQUOTA	Current process JPI\$_ASTLM  Current process JPI\$_BIOLM  Current process JPI\$_BYTLM  Current process JPI\$_CPULM



	PQL-PGFLQUOTA PQL-PRCLM PQL-TQELM	Current process JPI\$_DIOLM  Current process JPI\$_ENQLM  Current process JPI\$_FILLM  Current process JPI\$_JTQUOTA  Current process JPI\$_PGFLQUOTA  Current process JPI\$_PRCLM  Current process JPI\$_TQELM
DOMAINNAME	PQL-ASTLM PQL-BIOLM PQL-BYTLM PQL-CPULM PQL-DIOLM PQL-ENQLM PQL-FILLM PQL-JTQUOTA PQL-PGFLQUOTA PQL-PRCLM PQL-TQELM	Current process JPI\$_ASTLM  Current process JPI\$_BIOLM  Current process JPI\$_BYTLM  Current process JPI\$_CPULM  Current process JPI\$_DIOLM  Current process JPI\$_ENQLM  Current process JPI\$_FILLM  Current process JPI\$_JTQUOTA  Current process JPI\$_PGFLQUOTA  Current process JPI\$_PRCLM  Current process JPI\$_TQELM
ECHO	PQL-ASTLM PQL-BIOLM PQL-BYTLM PQL-CPULM PQL-DIOLM PQL-ENQLM PQL-FILLM PQL-JTQUOTA PQL-PGFLQUOTA PQL-PRCLM PQL-TQELM	Current process JPI\$_ASTLM  Current process JPI\$_BIOLM  Current process JPI\$_BYTLM  Current process JPI\$_CPULM  Current process JPI\$_DIOLM  Current process JPI\$_ENQLM

		<p>Current process JPI\$_FILLM</p> <p>Current process JPI\$_JTQUOTA</p> <p>Current proces JPI\$_PGFLQUOTA</p> <p>Current process JPI\$_PRCLM</p> <p>Current process JPI\$_TQELM</p>
FINGER	<p>PQL-ASTLM</p> <p>PQL-BIOLM</p> <p>PQL-BYTLM</p> <p>PQL-CPULM</p> <p>PQL-DIOLM</p> <p>PQL-ENQLM</p> <p>PQL-FILLM</p> <p>PQL-JTQUOTA</p> <p>PQL-PGFLQUOTA</p> <p>PQL-PRCLM</p> <p>PQL-TQELM</p>	<p>Current process JPI\$_ASTLM</p> <p>Current process JPI\$_BIOLM</p> <p>Current process JPI\$_BYTLM</p> <p>Current process JPI\$_CPULM</p> <p>Current process JPI\$_DIOLM</p> <p>Current process JPI\$_ENQLM</p> <p>Current process JPI\$_FILLM</p> <p>Current process JPI\$_JTQUOTA</p> <p>Current proces JPI\$_PGFLQUOTA</p> <p>Current process JPI\$_PRCLM</p> <p>Current process JPI\$_TQELM</p>
FONTSERVER	<p>PQL-ASTLM</p> <p>PQL-BIOLM</p> <p>PQL-BYTLM</p> <p>PQL-CPULM</p> <p>PQL-DIOLM</p> <p>PQL-ENQLM</p> <p>PQL-FILLM</p> <p>PQL-JTQUOTA</p> <p>PQL-PGFLQUOTA</p> <p>PQL-PRCLM</p> <p>PQL-TQELM</p>	<p>1000</p> <p>500</p> <p>100000</p> <p>Current process JPI\$_CPULM</p> <p>500</p> <p>500</p> <p>500</p> <p>4096</p>

		65536 Current process JPI\$_PRCLM 500
FTP	PQL-ASTLM PQL-BIOLM PQL-BYTLM PQL-CPULM PQL-DIOLM PQL-ENQLM PQL-FILLM PQL-JTQUOTA PQL-PGFLQUOTA PQL-PRCLM PQL-TQELM	Current process JPI\$_ASTLM 30 Current process JPI\$_BYTLM Current process JPI\$_CPULM Current process JPI\$_DIOLM Current process JPI\$_ENQLM Current process JPI\$_FILLM Current process JPI\$_JTQUOTA Current proces JPI\$_PGFLQUOTA Current process JPI\$_PRCLM Current process JPI\$_TQELM
GATED	PQL-ASTLM PQL-BIOLM PQL-BYTLM PQL-CPULM PQL-DIOLM PQL-ENQLM PQL-FILLM PQL-JTQUOTA PQL-PGFLQUOTA PQL-PRCLM PQL-TQELM	250 150 100000 0 150 500 100 4096 65536 0

		100
KTELNET	PQL-ASTLM PQL-BIOLM PQL-BYTLM PQL-CPULM PQL-DIOLM PQL-ENQLM PQL-FILLM PQL-JTQUOTA PQL-PGFLQUOTA PQL-PRCLM PQL-TQELM	250 500 100000 0 150 500 100 4096 65536 0 100
LLMR	PQL-ASTLM PQL-BIOLM PQL-BYTLM PQL-CPULM PQL-DIOLM PQL-ENQLM PQL-FILLM PQL-JTQUOTA PQL-PGFLQUOTA PQL-PRCLM PQL-TQELM	250 150 100000 0 Current process JPI\$_DIOLM Current process JPI\$_ENQLM 100 4096 65536 0 100

LPD	PQL-ASTLM PQL-BIOLM PQL-BYTLM PQL-CPULM PQL-DIOLM PQL-ENQLM PQL-FILLM PQL-JTQUOTA PQL-PGFLQUOTA PQL-PRCLM PQL-TQELM	Current process JPI\$_ASTLM  Current process JPI\$_BIOLM  Current process JPI\$_BYTLM  Current process JPI\$_CPULM  Current process JPI\$_DIOLM  Current process JPI\$_ENQLM  Current process JPI\$_FILLM  Current process JPI\$_JTQUOTA  Current process JPI\$_PGFLQUOTA  Current process JPI\$_PRCLM  Current process JPI\$_TQELM
NAMED	PQL-ASTLM PQL-BIOLM PQL-BYTLM PQL-CPULM PQL-DIOLM PQL-ENQLM PQL-FILLM PQL-JTQUOTA PQL-PGFLQUOTA PQL-PRCLM PQL-TQELM	250  Current process JPI\$_BIOLM  100000  0  Current process JPI\$_DIOLM  500  100  4096  65536  0  100
NETCONTROL	PQL-ASTLM PQL-BIOLM PQL-BYTLM PQL-CPULM	Current process JPI\$_ASTLM  Current process JPI\$_BIOLM

	PQL-DIOLM PQL-ENQLM PQL-FILLM PQL-JTQUOTA PQL-PGFLQUOTA PQL-PRCLM PQL-TQELM	Current process JPI\$_BYTLM  Current process JPI\$_CPULM  Current process JPI\$_DIOLM  Current process JPI\$_ENQLM  Current process JPI\$_FILLM  Current process JPI\$_JTQUOTA  Current process JPI\$_PGFLQUOTA  Current process JPI\$_PRCLM  Current process JPI\$_TQELM
NETSTAT	PQL-ASTLM PQL-BIOLM PQL-BYTLM PQL-CPULM PQL-DIOLM PQL-ENQLM PQL-FILLM PQL-JTQUOTA PQL-PGFLQUOTA PQL-PRCLM PQL-TQELM	Current process JPI\$_ASTLM  Current process JPI\$_BIOLM  Current process JPI\$_BYTLM  Current process JPI\$_CPULM  Current process JPI\$_DIOLM  Current process JPI\$_ENQLM  Current process JPI\$_FILLM  Current process JPI\$_JTQUOTA  Current process JPI\$_PGFLQUOTA  Current process JPI\$_PRCLM  Current process JPI\$_TQELM
NFS	PQL-ASTLM PQL-BIOLM PQL-BYTLM PQL-CPULM PQL-DIOLM PQL-ENQLM PQL-FILLM PQL-JTQUOTA	1000  500  100000  Current process JPI\$_CPULM

	PQL-PGFLQUOTA PQL-PRCLM PQL-TQELM	DIOLM  500  500  4096  65536  Current process JPI\$_PRCLM  500
NTALK	PQL-ASTLM PQL-BIOLM PQL-BYTLM PQL-CPULM PQL-DIOLM PQL-ENQLM PQL-FILLM PQL-JTQUOTA PQL-PGFLQUOTA PQL-PRCLM PQL-TQELM	Current process JPI\$_ASTLM  Current process JPI\$_BIOLM  Current process JPI\$_BYTLM  Current process JPI\$_CPULM  Current process JPI\$_DIOLM  Current process JPI\$_ENQLM  Current process JPI\$_FILLM  Current process JPI\$_JTQUOTA  Current proces JPI\$_PGFLQUOTA  Current process JPI\$_PRCLM  Current process JPI\$_TQELM
NTP	PQL-ASTLM PQL-BIOLM PQL-BYTLM PQL-CPULM PQL-DIOLM PQL-ENQLM PQL-FILLM PQL-JTQUOTA PQL-PGFLQUOTA PQL-PRCLM PQL-TQELM	Current process JPI\$_ASTLM  500  100000  Current process JPI\$_CPULM  500  500

		<p>100</p> <p>4096</p> <p>65536</p> <p>Current process JPI\$_PRCLM</p> <p>100</p>
PCNFSD	<p>PQL-ASTLM</p> <p>PQL-BIOLM</p> <p>PQL-BYTLM</p> <p>PQL-CPULM</p> <p>PQL-DIOLM</p> <p>PQL-ENQLM</p> <p>PQL-FILLM</p> <p>PQL-JTQUOTA</p> <p>PQL-PGFLQUOTA</p> <p>PQL-PRCLM</p> <p>PQL-TQELM</p>	<p>Current process JPI\$_ASTLM</p> <p>Current process JPI\$_BIOLM</p> <p>Current process JPI\$_BYTLM</p> <p>Current process JPI\$_CPULM</p> <p>Current process JPI\$_DIOLM</p> <p>Current process JPI\$_ENQLM</p> <p>Current process JPI\$_FILLM</p> <p>Current process JPI\$_JTQUOTA</p> <p>Current process JPI\$_PGFLQUOTA</p> <p>Current process JPI\$_PRCLM</p> <p>Current process JPI\$_TQELM</p>
POP2	<p>PQL-ASTLM</p> <p>PQL-BIOLM</p> <p>PQL-BYTLM</p> <p>PQL-CPULM</p> <p>PQL-DIOLM</p> <p>PQL-ENQLM</p> <p>PQL-FILLM</p> <p>PQL-JTQUOTA</p> <p>PQL-PGFLQUOTA</p> <p>PQL-PRCLM</p> <p>PQL-TQELM</p>	<p>Current process JPI\$_ASTLM</p> <p>Current process JPI\$_BIOLM</p> <p>Current process JPI\$_BYTLM</p> <p>Current process JPI\$_CPULM</p> <p>Current process JPI\$_DIOLM</p> <p>Current process JPI\$_ENQLM</p> <p>Current process JPI\$_FILLM</p> <p>Current process JPI\$_JTQUOTA</p>



		<p>Current process JPI\$_PGFLQUOTA</p> <p>Current process JPI\$_PRCLM</p> <p>Current process JPI\$_TQELM</p>
POP3	<p>PQL-ASTLM</p> <p>PQL-BIOLM</p> <p>PQL-BYTLM</p> <p>PQL-CPULM</p> <p>PQL-DIOLM</p> <p>PQL-ENQLM</p> <p>PQL-FILLM</p> <p>PQL-JTQUOTA</p> <p>PQL-PGFLQUOTA</p> <p>PQL-PRCLM</p> <p>PQL-TQELM</p>	<p>Current process JPI\$_ASTLM</p> <p>Current process JPI\$_BIOLM</p> <p>Current process JPI\$_BYTLM</p> <p>Current process JPI\$_CPULM</p> <p>Current process JPI\$_DIOLM</p> <p>Current process JPI\$_ENQLM</p> <p>Current process JPI\$_FILLM</p> <p>Current process JPI\$_JTQUOTA</p> <p>Current proces JPI\$_PGFLQUOTA</p> <p>Current process JPI\$_PRCLM</p> <p>Current process JPI\$_TQELM</p>
RACOON	<p>PQL-ASTLM</p> <p>PQL-BIOLM</p> <p>PQL-BYTLM</p> <p>PQL-CPULM</p> <p>PQL-DIOLM</p> <p>PQL-ENQLM</p> <p>PQL-FILLM</p> <p>PQL-JTQUOTA</p> <p>PQL-PGFLQUOTA</p> <p>PQL-PRCLM</p> <p>PQL-TQELM</p>	<p>250</p> <p>150</p> <p>100000</p> <p>0</p> <p>150</p> <p>500</p> <p>100</p> <p>4096</p> <p>65536</p> <p>0</p>

		100
RARP	PQL-ASTLM PQL-BIOLM PQL-BYTLM PQL-CPULM PQL-DIOLM PQL-ENQLM PQL-FILLM PQL-JTQUOTA PQL-PGFLQUOTA PQL-PRCLM PQL-TQELM	Current process JPI\$_ASTLM  Current process JPI\$_BIOLM  Current process JPI\$_BYTLM  Current process JPI\$_CPULM  Current process JPI\$_DIOLM  Current process JPI\$_ENQLM  Current process JPI\$_FILLM  Current process JPI\$_JTQUOTA  Current process JPI\$_PGFLQUOTA  Current process JPI\$_PRCLM  Current process JPI\$_TQELM
REXEC	PQL-ASTLM PQL-BIOLM PQL-BYTLM PQL-CPULM PQL-DIOLM PQL-ENQLM PQL-FILLM PQL-JTQUOTA PQL-PGFLQUOTA PQL-PRCLM PQL-TQELM	Current process JPI\$_ASTLM  Current process JPI\$_BIOLM  Current process JPI\$_BYTLM  Current process JPI\$_CPULM  Current process JPI\$_DIOLM  Current process JPI\$_ENQLM  Current process JPI\$_FILLM  Current process JPI\$_JTQUOTA  Current proces JPI\$_PGFLQUOTA  Current process JPI\$_PRCLM  Current process JPI\$_TQELM

RLOGIN	PQL-ASTLM PQL-BIOLM PQL-BYTLM PQL-CPULM PQL-DIOLM PQL-ENQLM PQL-FILLM PQL-JTQUOTA PQL-PGFLQUOTA PQL-PRCLM PQL-TQELM	Current process JPI\$_ASTLM  Current process JPI\$_BIOLM  Current process JPI\$_BYTLM  Current process JPI\$_CPULM  Current process JPI\$_DIOLM  Current process JPI\$_ENQLM  Current process JPI\$_FILLM  Current process JPI\$_JTQUOTA  Current process JPI\$_PGFLQUOTA  Current process JPI\$_PRCLM  Current process JPI\$_TQELM
RPCBOOTPARAM	PQL-ASTLM PQL-BIOLM PQL-BYTLM PQL-CPULM PQL-DIOLM PQL-ENQLM PQL-FILLM PQL-JTQUOTA PQL-PGFLQUOTA PQL-PRCLM PQL-TQELM	Current process JPI\$_ASTLM  Current process JPI\$_BIOLM  Current process JPI\$_BYTLM  Current process JPI\$_CPULM  Current process JPI\$_DIOLM  Current process JPI\$_ENQLM  Current process JPI\$_FILLM  Current process JPI\$_JTQUOTA  Current process JPI\$_PGFLQUOTA  Current process JPI\$_PRCLM  Current process JPI\$_TQELM
RPCLOCKMGR	PQL-ASTLM PQL-BIOLM PQL-BYTLM PQL-CPULM	Current process JPI\$_ASTLM  Current process JPI\$_BIOLM

	PQL-DIOLM PQL-ENQLM PQL-FILLM PQL-JTQUOTA PQL-PGFLQUOTA PQL-PRCLM PQL-TQELM	Current process JPI\$_BYTLM  Current process JPI\$_CPULM  Current process JPI\$_DIOLM  Current process JPI\$_ENQLM  Current process JPI\$_FILLM  Current process JPI\$_JTQUOTA  Current process JPI\$_PGFLQUOTA  Current process JPI\$_PRCLM  Current process JPI\$_TQELM
RPCMOUNT	PQL-ASTLM PQL-BIOLM PQL-BYTLM PQL-CPULM PQL-DIOLM PQL-ENQLM PQL-FILLM PQL-JTQUOTA PQL-PGFLQUOTA PQL-PRCLM PQL-TQELM	Current process JPI\$_ASTLM  Current process JPI\$_BIOLM  Current process JPI\$_BYTLM  Current process JPI\$_CPULM  Current process JPI\$_DIOLM  Current process JPI\$_ENQLM  Current process JPI\$_FILLM  Current process JPI\$_JTQUOTA  Current process JPI\$_PGFLQUOTA  Current process JPI\$_PRCLM  Current process JPI\$_TQELM
RPCPORTMAP	PQL-ASTLM PQL-BIOLM PQL-BYTLM PQL-CPULM PQL-DIOLM PQL-ENQLM PQL-FILLM PQL-JTQUOTA	Current process JPI\$_ASTLM  Current process JPI\$_BIOLM  Current process JPI\$_BYTLM  Current process JPI\$_CPULM

	PQL-PGFLQUOTA PQL-PRCLM PQL-TQELM	Current process JPI\$_DIOLM  Current process JPI\$_ENQLM  Current process JPI\$_FILLM  Current process JPI\$_JTQUOTA  Current process JPI\$_PGFLQUOTA  Current process JPI\$_PRCLM  Current process JPI\$_TQELM
RPCQUOTAD	PQL-ASTLM PQL-BIOLM PQL-BYTLM PQL-CPULM PQL-DIOLM PQL-ENQLM PQL-FILLM PQL-JTQUOTA PQL-PGFLQUOTA PQL-PRCLM PQL-TQELM	Current process JPI\$_ASTLM  Current process JPI\$_BIOLM  Current process JPI\$_BYTLM  Current process JPI\$_CPULM  Current process JPI\$_DIOLM  Current process JPI\$_ENQLM  Current process JPI\$_FILLM  Current process JPI\$_JTQUOTA  Current process JPI\$_PGFLQUOTA  Current process JPI\$_PRCLM  Current process JPI\$_TQELM
RPCSTATUS	PQL-ASTLM PQL-BIOLM PQL-BYTLM PQL-CPULM PQL-DIOLM PQL-ENQLM PQL-FILLM PQL-JTQUOTA PQL-PGFLQUOTA PQL-PRCLM PQL-TQELM	Current process JPI\$_ASTLM  Current process JPI\$_BIOLM  Current process JPI\$_BYTLM  Current process JPI\$_CPULM  Current process JPI\$_DIOLM  Current process JPI\$_ENQLM

		<p>Current process JPI\$_FILLM</p> <p>Current process JPI\$_JTQUOTA</p> <p>Current proces JPI\$_PGFLQUOTA</p> <p>Current process JPI\$_PRCLM</p> <p>Current process JPI\$_TQELM</p>
RSHELL	<p>PQL-ASTLM</p> <p>PQL-BIOLM</p> <p>PQL-BYTLM</p> <p>PQL-CPULM</p> <p>PQL-DIOLM</p> <p>PQL-ENQLM</p> <p>PQL-FILLM</p> <p>PQL-JTQUOTA</p> <p>PQL-PGFLQUOTA</p> <p>PQL-PRCLM</p> <p>PQL-TQELM</p>	<p>Current process JPI\$_ASTLM</p> <p>Current process JPI\$_BIOLM</p> <p>Current process JPI\$_BYTLM</p> <p>Current process JPI\$_CPULM</p> <p>Current process JPI\$_DIOLM</p> <p>Current process JPI\$_ENQLM</p> <p>Current process JPI\$_FILLM</p> <p>Current process JPI\$_JTQUOTA</p> <p>Current proces JPI\$_PGFLQUOTA</p> <p>Current process JPI\$_PRCLM</p> <p>Current process JPI\$_TQELM</p>
RUSERS	<p>PQL-ASTLM</p> <p>PQL-BIOLM</p> <p>PQL-BYTLM</p> <p>PQL-CPULM</p> <p>PQL-DIOLM</p> <p>PQL-ENQLM</p> <p>PQL-FILLM</p> <p>PQL-JTQUOTA</p> <p>PQL-PGFLQUOTA</p> <p>PQL-PRCLM</p> <p>PQL-TQELM</p>	<p>Current process JPI\$_ASTLM</p> <p>Current process JPI\$_BIOLM</p> <p>Current process JPI\$_BYTLM</p> <p>Current process JPI\$_CPULM</p> <p>Current process JPI\$_DIOLM</p> <p>Current process JPI\$_ENQLM</p> <p>Current process JPI\$_FILLM</p> <p>Current process JPI\$_JTQUOTA</p>

		<p>Current process JPI\$_PGFLQUOTA</p> <p>Current process JPI\$_PRCLM</p> <p>Current process JPI\$_TQELM</p>
RWALL	<p>PQL-ASTLM</p> <p>PQL-BIOLM</p> <p>PQL-BYTLM</p> <p>PQL-CPULM</p> <p>PQL-DIOLM</p> <p>PQL-ENQLM</p> <p>PQL-FILLM</p> <p>PQL-JTQUOTA</p> <p>PQL-PGFLQUOTA</p> <p>PQL-PRCLM</p> <p>PQL-TQELM</p>	<p>Current process JPI\$_ASTLM</p> <p>Current process JPI\$_BIOLM</p> <p>Current process JPI\$_BYTLM</p> <p>Current process JPI\$_CPULM</p> <p>Current process JPI\$_DIOLM</p> <p>Current process JPI\$_ENQLM</p> <p>Current process JPI\$_FILLM</p> <p>Current process JPI\$_JTQUOTA</p> <p>Current process JPI\$_PGFLQUOTA</p> <p>Current process JPI\$_PRCLM</p> <p>Current process JPI\$_TQELM</p>
SMTF	<p>PQL-ASTLM</p> <p>PQL-BIOLM</p> <p>PQL-BYTLM</p> <p>PQL-CPULM</p> <p>PQL-DIOLM</p> <p>PQL-ENQLM</p> <p>PQL-FILLM</p> <p>PQL-JTQUOTA</p> <p>PQL-PGFLQUOTA</p> <p>PQL-PRCLM</p> <p>PQL-TQELM</p>	<p>Current process JPI\$_ASTLM</p> <p>Current process JPI\$_BIOLM</p> <p>Current process JPI\$_BYTLM</p> <p>Current process JPI\$_CPULM</p> <p>Current process JPI\$_DIOLM</p> <p>Current process JPI\$_ENQLM</p> <p>Current process JPI\$_FILLM</p> <p>Current process JPI\$_JTQUOTA</p> <p>Current process JPI\$_PGFLQUOTA</p> <p>Current process JPI\$_PRCLM</p>

		Current process JPI\$_TQELM
SMUX	PQL-ASTLM PQL-BIOLM PQL-BYTLM PQL-CPULM PQL-DIOLM PQL-ENQLM PQL-FILLM PQL-JTQUOTA PQL-PGFLQUOTA PQL-PRCLM PQL-TQELM	Current process JPI\$_ASTLM Current process JPI\$_BIOLM Current process JPI\$_BYTLM Current process JPI\$_CPULM Current process JPI\$_DIOLM Current process JPI\$_ENQLM Current process JPI\$_FILLM Current process JPI\$_JTQUOTA Current process JPI\$_PGFLQUOTA Current process JPI\$_PRCLM Current process JPI\$_TQELM
SNMP	PQL-ASTLM PQL-BIOLM PQL-BYTLM PQL-CPULM PQL-DIOLM PQL-ENQLM PQL-FILLM PQL-JTQUOTA PQL-PGFLQUOTA PQL-PRCLM PQL-TQELM	250 150 100000 0 150 500 100 4096 65536 0 100



SSH	PQL-ASTLM PQL-BIOLM PQL-BYTLM PQL-CPULM PQL-DIOLM PQL-ENQLM PQL-FILLM PQL-JTQUOTA PQL-PGFLQUOTA PQL-PRCLM PQL-TQELM	250  500  100000  0  150  500  100  4096  65536  0  100
SYSTAT	PQL-ASTLM PQL-BIOLM PQL-BYTLM PQL-CPULM PQL-DIOLM PQL-ENQLM PQL-FILLM PQL-JTQUOTA PQL-PGFLQUOTA PQL-PRCLM PQL-TQELM	Current process JPI\$_ASTLM  Current process JPI\$_BIOLM  Current process JPI\$_BYTLM  Current process JPI\$_CPULM  Current process JPI\$_DIOLM  Current process JPI\$_ENQLM  Current process JPI\$_FILLM  Current process JPI\$_JTQUOTA  Current proces JPI\$_PGFLQUOTA  Current process JPI\$_PRCLM  Current process JPI\$_TQELM
TALK	PQL-ASTLM PQL-BIOLM PQL-BYTLM PQL-CPULM	Current process JPI\$_ASTLM  Current process JPI\$_BIOLM

	PQL-DIOLM PQL-ENQLM PQL-FILLM PQL-JTQUOTA PQL-PGFLQUOTA PQL-PRCLM PQL-TQELM	Current process JPI\$_BYTLM  Current process JPI\$_CPULM  Current process JPI\$_DIOLM  Current process JPI\$_ENQLM  Current process JPI\$_FILLM  Current process JPI\$_JTQUOTA  Current process JPI\$_PGFLQUOTA  Current process JPI\$_PRCLM  Current process JPI\$_TQELM
TELNET	PQL-ASTLM PQL-BIOLM PQL-BYTLM PQL-CPULM PQL-DIOLM PQL-ENQLM PQL-FILLM PQL-JTQUOTA PQL-PGFLQUOTA PQL-PRCLM PQL-TQELM	Current process JPI\$_ASTLM  Current process JPI\$_BIOLM  Current process JPI\$_BYTLM  Current process JPI\$_CPULM  Current process JPI\$_DIOLM  Current process JPI\$_ENQLM  Current process JPI\$_FILLM  Current process JPI\$_JTQUOTA  Current process JPI\$_PGFLQUOTA  Current process JPI\$_PRCLM  Current process JPI\$_TQELM
TFTP	PQL-ASTLM PQL-BIOLM PQL-BYTLM PQL-CPULM PQL-DIOLM PQL-ENQLM PQL-FILLM PQL-JTQUOTA	Current process JPI\$_ASTLM  Current process JPI\$_BIOLM  Current process JPI\$_BYTLM  Current process JPI\$_CPULM

	PQL-PGFLQUOTA PQL-PRCLM PQL-TQELM	Current process JPI\$_DIOLM Current process JPI\$_ENQLM Current process JPI\$_FILLM Current process JPI\$_JTQUOTA Current process JPI\$_PGFLQUOTA Current process JPI\$_PRCLM Current process JPI\$_TQELM
TIME	PQL-ASTLM PQL-BIOLM PQL-BYTLM PQL-CPULM PQL-DIOLM PQL-ENQLM PQL-FILLM PQL-JTQUOTA PQL-PGFLQUOTA PQL-PRCLM PQL-TQELM	Current process JPI\$_ASTLM Current process JPI\$_BIOLM Current process JPI\$_BYTLM Current process JPI\$_CPULM Current process JPI\$_DIOLM Current process JPI\$_ENQLM Current process JPI\$_FILLM Current process JPI\$_JTQUOTA Current process JPI\$_PGFLQUOTA Current process JPI\$_PRCLM Current process JPI\$_TQELM
UCXQIO	PQL-ASTLM PQL-BIOLM PQL-BYTLM PQL-CPULM PQL-DIOLM PQL-ENQLM PQL-FILLM PQL-JTQUOTA PQL-PGFLQUOTA PQL-PRCLM PQL-TQELM	Current process JPI\$_ASTLM Current process JPI\$_BIOLM Current process JPI\$_BYTLM Current process JPI\$_CPULM Current process JPI\$_DIOLM Current process JPI\$_ENQLM

		<p>Current process JPI\$_FILLM</p> <p>Current process JPI\$_JTQUOTA</p> <p>Current proces JPI\$_PGFLQUOTA</p> <p>Current process JPI\$_PRCLM</p> <p>Current process JPI\$_TQELM</p>
UDPCHARGEN	<p>PQL-ASTLM</p> <p>PQL-BIOLM</p> <p>PQL-BYTLM</p> <p>PQL-CPULM</p> <p>PQL-DIOLM</p> <p>PQL-ENQLM</p> <p>PQL-FILLM</p> <p>PQL-JTQUOTA</p> <p>PQL-PGFLQUOTA</p> <p>PQL-PRCLM</p> <p>PQL-TQELM</p>	<p>Current process JPI\$_ASTLM</p> <p>Current process JPI\$_BIOLM</p> <p>Current process JPI\$_BYTLM</p> <p>Current process JPI\$_CPULM</p> <p>Current process JPI\$_DIOLM</p> <p>Current process JPI\$_ENQLM</p> <p>Current process JPI\$_FILLM</p> <p>Current process JPI\$_JTQUOTA</p> <p>Current proces JPI\$_PGFLQUOTA</p> <p>Current process JPI\$_PRCLM</p> <p>Current process JPI\$_TQELM</p>
UDPDAYTIME	<p>PQL-ASTLM</p> <p>PQL-BIOLM</p> <p>PQL-BYTLM</p> <p>PQL-CPULM</p> <p>PQL-DIOLM</p> <p>PQL-ENQLM</p> <p>PQL-FILLM</p> <p>PQL-JTQUOTA</p> <p>PQL-PGFLQUOTA</p> <p>PQL-PRCLM</p> <p>PQL-TQELM</p>	<p>Current process JPI\$_ASTLM</p> <p>Current process JPI\$_BIOLM</p> <p>Current process JPI\$_BYTLM</p> <p>Current process JPI\$_CPULM</p> <p>Current process JPI\$_DIOLM</p> <p>Current process JPI\$_ENQLM</p> <p>Current process JPI\$_FILLM</p> <p>Current process JPI\$_JTQUOTA</p>

		<p>Current process JPI\$_PGFLQUOTA</p> <p>Current process JPI\$_PRCLM</p> <p>Current process JPI\$_TQELM</p>
UDPDISCARD	<p>PQL-ASTLM</p> <p>PQL-BIOLM</p> <p>PQL-BYTLM</p> <p>PQL-CPULM</p> <p>PQL-DIOLM</p> <p>PQL-ENQLM</p> <p>PQL-FILLM</p> <p>PQL-JTQUOTA</p> <p>PQL-PGFLQUOTA</p> <p>PQL-PRCLM</p> <p>PQL-TQELM</p>	<p>Current process JPI\$_ASTLM</p> <p>Current process JPI\$_BIOLM</p> <p>Current process JPI\$_BYTLM</p> <p>Current process JPI\$_CPULM</p> <p>Current process JPI\$_DIOLM</p> <p>Current process JPI\$_ENQLM</p> <p>Current process JPI\$_FILLM</p> <p>Current process JPI\$_JTQUOTA</p> <p>Current proces JPI\$_PGFLQUOTA</p> <p>Current process JPI\$_PRCLM</p> <p>Current process JPI\$_TQELM</p>
UDPECHO	<p>PQL-ASTLM</p> <p>PQL-BIOLM</p> <p>PQL-BYTLM</p> <p>PQL-CPULM</p> <p>PQL-DIOLM</p> <p>PQL-ENQLM</p> <p>PQL-FILLM</p> <p>PQL-JTQUOTA</p> <p>PQL-PGFLQUOTA</p> <p>PQL-PRCLM</p> <p>PQL-TQELM</p>	<p>Current process JPI\$_ASTLM</p> <p>Current process JPI\$_BIOLM</p> <p>Current process JPI\$_BYTLM</p> <p>Current process JPI\$_CPULM</p> <p>Current process JPI\$_DIOLM</p> <p>Current process JPI\$_ENQLM</p> <p>Current process JPI\$_FILLM</p> <p>Current process JPI\$_JTQUOTA</p> <p>Current proces JPI\$_PGFLQUOTA</p> <p>Current process JPI\$_PRCLM</p>

		Current process JPI\$_TQELM
UDPTIME	PQL-ASTLM PQL-BIOLM PQL-BYTLM PQL-CPULM PQL-DIOLM PQL-ENQLM PQL-FILLM PQL-JTQUOTA PQL-PGFLQUOTA PQL-PRCLM PQL-TQELM	Current process JPI\$_ASTLM Current process JPI\$_BIOLM Current process JPI\$_BYTLM Current process JPI\$_CPULM Current process JPI\$_DIOLM Current process JPI\$_ENQLM Current process JPI\$_FILLM Current process JPI\$_JTQUOTA Current process JPI\$_PGFLQUOTA Current process JPI\$_PRCLM Current process JPI\$_TQELM
VIADENET	PQL-ASTLM PQL-BIOLM PQL-BYTLM PQL-CPULM PQL-DIOLM PQL-ENQLM PQL-FILLM PQL-JTQUOTA PQL-PGFLQUOTA PQL-PRCLM PQL-TQELM	Current process JPI\$_ASTLM Current process JPI\$_BIOLM Current process JPI\$_BYTLM Current process JPI\$_CPULM Current process JPI\$_DIOLM Current process JPI\$_ENQLM Current process JPI\$_FILLM Current process JPI\$_JTQUOTA Current process JPI\$_PGFLQUOTA Current process JPI\$_PRCLM Current process JPI\$_TQELM

XDM	PQL-ASTLM	1000
	PQL-BIOLM	
	PQL-BYTLM	500
	PQL-CPULM	
	PQL-DIOLM	100000
	PQL-ENQLM	
	PQL-FILLM	Current process JPI\$_CPULM
	PQL-JTQUOTA	
	PQL-PGFLQUOTA	500
	PQL-PRCLM	
	PQL-TQELM	500
		500
		4096
		65536
	Current process JPI\$_PRCLM	
	500	

# Appendix B. DNSSEC

## DNSSEC

Cryptographic authentication of DNS information is possible through the DNS Security (DNSSEC-bis) extensions, defined in RFC 4033, RFC 4034, and RFC 4035. This section describes the creation and use of DNSSEC signed zones.

In order to setup a DNSSEC secure zone, there are a series of steps which must be followed. BIND9 ships with several tools that are used in this process, which are explained in more detail below. In all cases, the `-h` option prints a full list of parameters.

**NOTE:** For use with MultiNet, define symbols for each of the tools, and call the symbol from the command line. For example, to use the `dnssec-keygen` tool, a symbol could be created as follows:

```
$ keygen := $multinet:dnssec-keygen
$ keygen -h
```

There must also be communication with the administrators of the parent and/or child zone to transmit keys. A zone's security status must be indicated by the parent zone for a DNSSEC capable resolver to trust its data. This is done through the presence or absence of a DS record at the delegation point.

For other servers to trust data in this zone, they must either be statically configured with this zone's zone key or the zone key of another zone above this one in the DNS tree.

## Generating Keys

The `dnssec-keygen` program is used to generate keys. A secure zone must contain one or more zone keys. The zone keys will 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, a name type of ZONE, and must be



usable for authentication. It is recommended that zone keys use a cryptographic algorithm designated as “mandatory to implement” by the IETF; currently the only one is RSASHA1. For convenience, run the `dnssec-keygen` tool in the directory the zone data files are located, so you won’t need to use full pathnames as arguments.

The following command will generate a 768-bit RSASHA1 key for the `child.example` zone, the symbol `keygen` has been created to refer to the `dnssec-keygen` executable:

```
$ keygen -a RSASHA1 -b 768 -n ZONE child-example
```

**NOTE:** File names specified with the tools must conform to OpenVMS naming conventions. Be aware of using multiple dots, etc. which will generate errors upon file creation.

Two output files will be produced: `Kchild-example-005-12345.key` and `Kchild-example-005-12345.private` (where 12345 is an example of a key tag). The key filenames contain the key name (`child-example.`), algorithm (3 is DSA, 1 is RSAMD5, 5 is RSASHA1, etc.), 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 for signature verification.

**CAUTION!** Always protect the `.private` key, anyone who knows it can forge signed zone data. The `.private` key file will be written readable and writable only by the user who runs it. Process Software recommends running the DNSSEC tools from a suitably privileged account.

To generate another key with the same properties (but with a different key tag), repeat the above command.

The `dnssec-keyfromlabel` program is used to get a key pair from a crypto hardware and build the key files. Its usage is similar to `dnssec-keygen`.

The public keys can be inserted into the zone file by pasting in their contents, or better yet by including the `.key` file using `$include` statements. For example, to insert the public key for `child-example`, add the following `$include` statement to the zone file:

```
$include Kchild-example-005-12345.key ;
```

The zone file (for examples in this Appendix, the file name is `zone.1`) may look like this:

```
$TTL 100
$ORIGIN child-example.
@      IN SOA  a.example. a.a.example. 1 360 36 60480 12
                NS      a.example.
                NS      b.example.
one    IN A    10.10.10.10
two    IN A    10.10.10.100
                MX      10 one.zz.example.
$include Kchild-example-005-12345.key ;
```

## Signing the Zone

With the key included in the zone file, use the `dnssec-signzone` program to sign the zone.

Any keyset files corresponding to secure subzones should be present. The zone signer will generate NSEC, NSEC3 and RRSIG records for the zone, as well as DS for the child zones if `-g` is specified. If `-g` is not specified, then DS RRsets for the secure child zones need to be added manually.

The following command signs the zone, assuming it is in a file called `zone.1`. By default, all zone keys which have an available private key are used to generate signatures. First define a symbol for `dnssec-signzone`:

```
$ signer := $multinet:dnssec-signzone
```

The `-o` option specifies the zone origin, the default is the zone file name.

```
$ signer -o child-example zone.1
```

**Note:** You may see the message `No self signing KSK found`. This is normal as no KSK (key signing key) has been generated at this point. Only a ZSK (zone signing key) is present.

One output file is produced: `zone.1_signed`. This file should be referenced by `named.conf` as the input file for the zone.

The output file, `zone.1_signed`, should look something like this:

```
; dnssec_signzone version 9.7.2-p3
```

```

child-example. 100      IN SOA  a.example. a.a.example. (
    1          ; serial
    360       ; refresh (6 minutes)
    36        ; retry (36 seconds)
    60480     ; expire (16 hours 48 minutes)
    12        ; minimum (12 seconds)
)
100      RRSIG  SOA 5 1 100 20110428114855 (
    20110329114855 36111 child-example.
    rWVs/euooBTVk0MzhxHQio61rDBhzAId13sV
    KXphVsA64bqyayhJcCfikmxww6vq6gG0W3mR
    z1tbIQ7znZ0SN90dsWhEcoEaEmm1Sl6hwsVY
    OzaYrN8HgahzcrNlsX5l )
100      NS     a.example.
100      NS     b.example.
100      RRSIG  NS 5 1 100 20110428114855 (
    20110329114855 36111 child-example.
    SOrA8BihARhE+SPl/iYjB8PTqk+8lc4sEE4b
    CYhgcF6d9VOZtCotQFUqVKrk65xoGqf60+9R
    kBjR6lsOwr6mqDVCiZzVnAy1frWD8T8q5HNK
    nzVR8gb7AXyPtbgKqOS3 )
12       NSEC   one.child-example. NS SOA RRSIG NSEC DNSKEY
12       RRSIG  NSEC 5 1 12 20110428114855 (
    20110329114855 36111 child-example.
    L0K9USccXSgO4iYBaXDOOrQ0zzrxVVREcWjAb
    DAeZqVec525V6kNIB5F2mCxjsJq1J5C40vr+
    lCqe/EGzjxplEzqq0nSN/fCtTgXqhLL6EfZx
    M1lvB5C+4K4hr20neVWy )
100      DNSKEY 256 3 5 (
    AwEAAcXIK+ljUWgMENcs9TUqnZGEFMOE5DBP
    WyQu5aIGSZqTTcvMWsaFtS7800LjapDB4kcs
    xwecfdA4I/0dUHPuHqmQREGfq/xstyxLPHKS
    MEkJthkVurf4MWzdX8dAVEd/GQ==
    ) ; key id = 36111
100      RRSIG  DNSKEY 5 1 100 20110428114855 (
    20110329114855 36111 child-example.
    O8t9100vLCSotc7mTG7iVr6fyeg7AA6ZuzHR
    Gfn0dbOFzZHGxSAj2pRXPz8FC/eYz+ngy6rK
    23UhdklmuJN35IEA+qkXBils7NJtEvaONoud
    1ANN6qQDtXyYFxnCuEN0 )
one.child-example. 100      IN A      10.10.10.10
100      RRSIG  A 5 2 100 20110428114855 (
    20110329114855 36111 child-example.
    o3TPUffd5dLuxoac0TVVsT8HU3MFoJtIbfXV
    apidfBY7IbxU6YWgPPwkYO1oKgJ3CnWmKTZQ
    sUB+QRE1VHn8GmPbyjbg9QfhIKZDEQyT2f7x
    41QDNznnKnJyYjhmbYcf )
12       NSEC   two.child-example. A RRSIG NSEC
12       RRSIG  NSEC 5 2 12 20110428114855 (
    20110329114855 36111 child-example.
    w3RXqBeiUk/njCh/nHg2s1hv9kYynGdRsp2A
    vYm8ahrq4pGv1DLr6uuwCT5vBfjor115ePBj

```

```

      jsIO3FLkWyO7miBpfiLLPa7umKSQLN0AZGIE
      /5Z7LSc80o2fzwqcBkub )
two.child-example.      100      IN A      10.10.10.100
100      RRSIG      A 5 2 100 20110428114855 (
      20110329114855 36111 child-example.
      jQAof31o6b04oOVlhLAt6NQkifz1l4qnfN4a
      viZiB0RmLYuRnNHFRApyZLkoI8PTgCuCdV/e
      colifFnXU9UauNnK/wQw8Djurvra/YMq8f5W
      ZZcOReQvZUoD8mS4C3ec )
100      MX      10 one.zz.example.
100      RRSIG      MX 5 2 100 20110428114855 (
      20110329114855 36111 child-example.
      hIQI20XS9qYdi5/3qMp1VeU0aQqBwQsugkw
      mCD9gY7BrpYjMeeg3XQHY0Qx7ElqLc9Q0F3C
      kC0ETM5CDnUAicXCy2TOc1DAKfSOYlKRnzVd
      a5LlFGymsi2gVyW7VssH )
12      NSEC      child-example. A MX RRSIG NSEC
12      RRSIG      NSEC 5 2 12 20110428114855 (
      20110329114855 36111 child-example.
      OhIM8y6IGXixOUtD+ZH/bicznRtX6YrdeXxg
      5bD3ROSUcfpCL5YAUxfk/B9nj2n10Stle88r
      O7EeMB2rSiAPqYW88ZbIXXhOHsE6z3ff7Plc
      B3pT56MBxUh5cm2WDYTL )

```

`dnssec-signzone` will also produce a `keyset` and `dsset` files and optionally a `dlvset` file. These are used to provide the parent zone administrators with the DNSKEYs (or their corresponding DS records) that are the secure entry point to the zone.

## Configuring Servers

To enable `named` to respond appropriately to DNS requests from DNSSEC aware clients, the option `dnssec-enable` must be set to `yes`. (This is the default setting.)

To enable `named` to validate answers from other servers, the `dnssec-enable` and `dnssec-validation` options must both be set to `yes`, and at least one trust anchor must be configured with a `trusted-keys` or `managed-keys` statement in `named.conf`.

`trusted-keys` are copies of DNSKEY RRs for zones that are used to form the first link in the cryptographic chain of trust. All keys listed in `trusted-keys` (and corresponding zones) are deemed to exist and only the listed keys will be used to validate the DNSKEY RRset that they are from.

`managed-keys` are trusted keys which are automatically kept up to date via RFC 5011 trust anchor maintenance.

After DNSSEC gets established, a typical DNSSEC configuration will look something like the following. It has one or more public keys for the root. This allows answers from outside the

organization to be validated. It will also have several keys for parts of the namespace the organization controls. These are here to ensure that named is immune to compromises in the DNSSEC components of the security of parent zones.

```
managed-keys {
/* Root Key */
    "." initial-key 257 3 3

    "BNY4wrWM1nCfJ+CXd0rVXyYmobt7sEEfK3clRbGaTwS
    JxrGkxJWoZu6I7PzJu/E9gx4UC1zGAHlXKdE4zYIprh
    aBKnvcC2U9mZhkDUpd1Vso/HAdjNe8LmMlnzY3zy2Xy
    4klWOADTPzSv9eamj8V18PHGjBLaVtYvk/ln5ZApjYg
                                hf+6fElrmLkdaz MQ2OCnACR817DF4BBa7UR/beDHyp
                                5iWTXWSi6XmoJLbG9Scqc7170KDq1vXR3M/1UUVRbke
                                g1IPJSidmK3ZyC1lh4XSKbje/45SKucHgnwU5jefMtg
                                66gKodQj+MiA21AfUve7u99WzTLzY3qlxDhxYQQ20FQ
                                97S+LKUTpQcq27R7AT3/V5hRQxScINqwcZ4jYqZD2fQ
                                dgxbcDTC1U0CRBdiieyLMNzXG3";

};

trusted-keys {
/* Key for our organization's forward zone */
example.net. 257 3 5
"AwEAAaxPMcR2x0HbQV4WeZB6oEDX+r0QM6
5KbhTjrw1ZaArmPhEZZe3Y9ifgEuq7vZ/z
GZUdEGNWY+JZzus0lUptwgjGwhUS1558Hb
4JKUbbOTcM8pwXlj0EiX3oDFVmjHO444gL
kBOUKUf/mC7HvfwYH/Be22GnClrinKJp10
g4yWzO9WglMk7jbfW33gUKvirTHr25GL7S
TQUzBb5Usxt8lgnyTUHs1t3JwCY5hKZ6Cq
FxmAVZP20igTixin/1LcrgX/KMEGd/biuv
F4qJCyduieHukuY3H4XMAcR+xia2nIUPvm
/oyWR8BW/hWdzOvnSCThlHf3xiYleDbt/o
1OTQ09A0=";

/* Key for our reverse zone. */

2.0.192.IN-ADDRPA.NET. 257 3 5
"AQOnS4xn/IgOUpBPJ3bogzwc
xOdNax071L18QqZnQQQAVVr+i
LhGTnNGp3HoWQLUIzKrJVZ3zg
gy3WwNT6kZo6c0tszYqbtvchm
gQC8CzKojM/W16i6MG/eafGU3
siaOdS0yOI6BgPsw+YZdzlYMa
IJGf4M4dyoKIhzdZyQ2bYQrjy
Q4LB01c7aOnsMyYKHHYeRvPxj
IQXmdqgOJGq+vsevG06zW+1xg
YJh9rCIfnm1GX/KMgxLPG2vXT
D/RnLX+D3T3UL7HJYHJhAZD5L
59VvjSPsZJHeDCUyWYrvPZesZ
DIRvhDD52SKvbheeTJU6Ehkz
```

```
ytNN2SN96QRk8j/iI8ib";  
};  
  
options { ...  
dnssec-enable yes;  
dnssec-validation yes;  
};
```

**NOTE:** None of the keys listed in this example are valid. In particular, the root key is not valid.

When DNSSEC validation is enabled and properly configured, the resolver will reject any answers from signed, secure zones which fail to validate, and will return `SERVFAIL` to the client.

Responses may fail to validate for any of several reasons, including missing, expired, or invalid signatures, a key which does not match the DS RRset in the parent zone, or an insecure response from a zone which, according to its parent, should have been secure.

**NOTE:** When the validator receives a response from an unsigned zone that has a signed parent, it must confirm with the parent that the zone was intentionally left unsigned. It does this by verifying, via signed and validated NSEC/NSEC3 records, that the parent zone contains no DS records for the child. If the validator can prove that the zone is insecure, then the response is accepted. However, if it cannot, then it must assume an insecure response to be a forgery; it rejects the response and logs an error. The logged error reads `insecurity proof failed and got insecure response; parent indicates it should be secure`.

## DNSSEC, DYNAMIC ZONES, AND AUTOMATIC SIGNING

As of BIND 9.7.0 it is possible to change a dynamic zone from insecure to signed and back again. A secure zone can use either NSEC or NSEC3 chains.

# Converting from insecure to secure

Changing a zone from insecure to secure can be done in two ways: using a dynamic DNS update, or the `auto-dnssec` zone option.

For either method, you need to configure `named` so that it can see the `K*` files which contain the public and private parts of the keys that will be used to sign the zone. These files will have been generated by `dnssec-keygen`. You can do this by placing them in the `key-directory`, as specified in `named.conf`:

```
zone example.net {
    type master;
    update-policy local;
    file "example.net";
    key-directory "multinet_common_root:[multinet]";
};
```

If one KSK and one ZSK DNSKEY key have been generated, this configuration will cause all records in the zone to be signed with the ZSK, and the DNSKEY RR set to be signed with the KSK as well. An NSEC chain will be generated as part of the initial signing process.

## Dynamic DNS update method

To insert the keys via dynamic update:

```
$ nsupdate := $multinet:nsupdate.exe
$ nsupdate
> ttl 3600
> update add example.net DNSKEY 256 3 7
AwEAAZn17pUF0KpbPA2c7Gz76Vb18v0teKT3EyAGfBfL8eQ8a135zz3Y
> update add example.net DNSKEY 257 3 7
AwEAAAd/7odU/64o2LGsifbLtQmtO8dFDtTAZXSX2+
> send
```

While the update request will complete almost immediately, the zone will not be completely signed until `named` has had time to walk the zone and generate the NSEC and RRSIG records. The NSEC record at the apex will be added last, to signal that there is a complete NSEC chain.

If you wish to sign using NSEC3 instead of NSEC, you should add an `NSEC3PARAM` record to the initial update request. If you wish the NSEC3 chain to have the `OPTOUT` bit set, set it in the `flags` field of the `NSEC3PARAM` record.

```
$ nsupdate
> ttl 3600
> update add example.net DNSKEY 256 3 7
AwEAAZn17pUF0KpbPA2c7Gz76Vb18v0teKT3EyAGfBfL8eQ8a135zz3Y
```

```
> update add example.net DNSKEY 257 3 7
AwEAAAd/7odU/64o2LGsifbLtQmtO8dFDtTAZXSX2+X3e/
> update add example.net NSEC3PARAM 1 1 100 1234567890
> send
```

Again, this update request will complete almost immediately; however, the record won't show up until named has had a chance to build/remove the relevant chain. A private type record will be created to record the state of the operation (see below for more details), and will be removed once the operation completes.

While the initial signing and NSEC/NSEC3 chain generation is happening, other updates are possible as well.

## Fully automatic zone signing

To enable automatic signing, add the `auto-dnssec` option to the zone statement in `named.conf`. `auto-dnssec` has two possible arguments: `allow` or `maintain`.

With `auto-dnssec allow`, `named` can search the key directory for keys matching the zone, insert them into the zone, and use them to sign the zone. It will do so only when it receives an `rndc sign zonenumber` or `rndc loadkeys zonenumber` command.

`auto-dnssec maintain` includes the above functionality, but will also automatically adjust the zone's DNSKEY records on schedule according to the keys' timing metadata. If keys are present in the key directory the first time the zone is loaded, it will be signed immediately, without waiting for an `rndc sign` or `rndc loadkeys` command. (Those commands can still be used when there are unscheduled key changes, however.)

Using the `auto-dnssec` option requires the zone to be configured to allow dynamic updates, by adding an `allow-update` or `update-policy` statement to the zone configuration. If this has not been done, the configuration will fail.

## Private-type records

The state of the signing process is signaled by private-type records (with a default type value of 65534). When signing is complete, these records will have a non-zero value for the final octet (for those records which have a non-zero initial octet).



The private type record format: If the first octet is non-zero then the record indicates that the zone needs to be signed with the key matching the record, or that all signatures that match the record should be removed.

- algorithm (octet 1)
- key id in network order (octet 2 and 3)
- removal flag (octet 4)
- complete flag (octet 5)

Only records flagged as “complete” can be removed via dynamic update. Attempts to remove other private type records will be silently ignored. If the first octet is zero (this is a reserved algorithm number that should never appear in a DNSKEY record) then the record indicates changes to the NSEC3 chains are in progress. The rest of the record contains an NSEC3PARAM record. The flag field tells what operation to perform based on the flag bits.

- 0x01 OPTOUT
- 0x80 CREATE
- 0x40 REMOVE
- 0x20 NONSEC

## DNSKEY rollovers

As within secure-to-secure conversions, rolling DNSSEC keys can be done in two ways: using a dynamic DNS update, or the `auto-dnssec` zone option.

### Dynamic DNS update method

To perform key rollovers via dynamic update, you need to add the  $K^*$  files for the new keys so that `named` can find them. You can then add the new DNSKEY RRs via dynamic update. `named` will then cause the zone to be signed with the new keys. When the signing is complete the private type records will be updated so that the last octet is non-zero.

If this is for a KSK you need to inform the parent and any trust anchor repositories of the new KSK.

You should then wait for the maximum TTL in the zone before removing the old DNSKEY. If it is a KSK that is being updated, you also need to wait for the DS RRset in the parent to be updated and its

TTL to expire. This ensures that all clients will be able to verify at least one signature when you remove the old DNSKEY.

The old DNSKEY can be removed via `UPDATE`. Take care to specify the correct key. `named` will clean out any signatures generated by the old key after the update completes.

## Automatic key rollovers

When a new key reaches its activation date (as set by `dnssec-keygen` or `dnssec-settime`), if the `auto-dnssec` zone option is set to `maintain`, `named` will automatically carry out the key roll over. If the key's algorithm has not previously been used to sign the zone, then the zone will be fully signed as quickly as possible. However, if the new key is replacing an existing key of the same algorithm, then the zone will be re-signed incrementally, with signatures from the old key being replaced with signatures from the new key as their signature validity periods expire. By default, this rollover completes in 30 days, after which it will be safe to remove the old key from the DNSKEY RRset.

## NSEC3PARAM rollovers via UPDATE

Add the new NSEC3PARAM record via dynamic update. When the new NSEC3 chain has been generated, the NSEC3PARAM flag field will be zero. At this point you can remove the old NSEC3PARAM record. The old chain will be removed after the update request completes.

## Converting from NSEC to NSEC3

To do this, you just need to add an NSEC3PARAM record. When the conversion is complete, the NSEC chain will have been removed and the NSEC3PARAM record will have a zero flag field. The NSEC3 chain will be generated before the NSEC chain is destroyed.

## Converting from NSEC3 to NSEC

To do this, use `nsupdate` to remove all NSEC3PARAM records with a zero flag field. The NSEC chain will be generated before the NSEC3 chain is removed.

## Converting from secure to insecure

To convert a signed zone to unsigned using dynamic DNS, delete all the `DNSKEY` records from the zone apex using `nsupdate`. All signatures, NSEC or NSEC3 chains, and associated NSEC3PARAM records will be removed automatically. This will take place after the update request completes.

This requires the `dnssec-secure-to-insecure` option to be set to `yes` in `named.conf`.

In addition, if the `auto-dnssec maintain zone` statement is used, it should be removed or changed to `allow` instead (or it will re-sign).

## Periodic re-signing

In any secure zone which supports dynamic updates, `named` will periodically re-sign RRsets which have not been re-signed as a result of some update action. The signature lifetimes will be adjusted so as to spread the re-sign load over time rather than all at once.

## NSEC3 and OPTOUT

`named` supports creating new NSEC3 chains where all the NSEC3 records in the zone have the same OPTOUT state. `named` also supports UPDATES to zones where the NSEC3 records in the chain have mixed OPTOUT state. `named` does not support changing the OPTOUT state of an individual NSEC3 record, the entire chain needs to be changed if the OPTOUT state of an individual NSEC3 needs to be changed.

## Dynamic Trust Anchor Management

MultiNet's version of BIND includes support for RFC 5011, dynamic trust anchor management. Using this feature allows `named` to keep track of changes to critical DNSSEC keys without any need for the operator to make changes to configuration files.

# Validating Resolver

To configure a validating resolver to use RFC 5011 to maintain a trust anchor, configure the trust anchor using a `managed-keys` statement.

## Authoritative Server

To set up an authoritative zone for RFC 5011 trust anchor maintenance, generate two (or more) key signing keys (KSKs) for the zone. Sign the zone with one of them; this is the “active” KSK. All KSK’s which do not sign the zone are “stand-by” keys.

Any validating resolver which is configured to use the active KSK as an RFC 5011-managed trust anchor will take note of the stand-by KSKs in the zone’s DNSKEY RRset, and store them for future reference. The resolver will recheck the zone periodically, and after 30 days, if the new key is still there, then the key will be accepted by the resolver as a valid trust anchor for the zone. Any time after this 30-day acceptance timer has completed, the active KSK can be revoked, and the zone can be “rolled over” to the newly accepted key.

The easiest way to place a stand-by key in a zone is to use the “smart signing” features of `dnssec-keygen` and `dnssec-signzone`. If the key has a publication date in the past, but an activation date which is unset or in the future, `dnssec-signzone -S` will include the DNSKEY record in the zone, but will not sign with it:

```
$ dnssec-keygen -K keys -f KSK -P now -A now+2y example.net
$ dnssec-signzone -S -K keys example.net
```

To revoke a key, the new command `dnssec-revoke` has been added. This adds the REVOKED bit to the key flags and re-generates the `K*.key` and `K*.private` files. After revoking the active key, the zone must be signed with both the revoked KSK and the new active KSK. (Smart signing takes care of this automatically.)

Once a key has been revoked and used to sign the DNSKEY RRset in which it appears, that key will never again be accepted as a valid trust anchor by the resolver. However, validation can proceed using the new active key (which had been accepted by the resolver when it was a stand-by key).

See RFC 5011 for more details on key rollover scenarios.

When a key has been revoked, its key ID changes, increasing by 128, and wrapping around at 65535. So, for example, the key `Kexample-net-005-10000` becomes `Kexample-net-005-10128`.

If two keys have ID’s exactly 128 apart, and one is revoked, then the two key ID’s will collide, causing several problems. To prevent this, `dnssec-keygen` will not generate a new key if another key is

present which may collide. This checking will only occur if the new keys are written to the same directory which holds all other keys in use for that zone.

Older versions of BIND9 did not have this precaution. Exercise caution if using key revocation on keys that were generated by previous releases, or if using keys stored in multiple directories or on multiple machines.

It is expected that a future release of BIND9 will address this problem in a different way, by storing revoked keys with their original unrevoked key ID's.

