Bull

System Management Guide
Communications and Networks

AIX

ORDER REFERENCE
86 A2 27EF 03

Bull

System Management Guide
Communications and Networks

AIX

Software

June 2003

BULL CEDOC
357 AVENUE PATTON
B.P.20845
49008 ANGERS CEDEX 01
FRANCE

ORDER REFERENCE
86 A2 27EF 03

Printed in France

Suggestions and criticisms concerning the form, content, and presentation of this book are invited. A form is provided at the end of this book for this purpose.

To order additional copies of this book or other Bull Technical Publications, you are invited to use the Ordering Form also provided at the end of this book.

Trademarks and Acknowledgements

We acknowledge the right of proprietors of trademarks mentioned in this book.

AIX® is a registered trademark of International Business Machines Corporation, and is being used under licence.

UNIX is a registered trademark in the United States of America and other countries licensed exclusively through the Open Group.

Linux is a registered trademark of Linus Torvalds.

The information in this document is subject to change without notice. Groupe Bull will not be liable for errors contained herein, or for incidental or consequential damages in connection with the use of this material.

About This Book

This book is for system administrators who maintain the operating system's network connections. Familiarity with the Base Operating System and the material covered in *AIX 5L Version 5.2 System Management Guide: Operating System and Devices* and *AIX 5L Version 5.2 System User's Guide: Communications and Networks* is necessary.

Beginning with the AIX 5.2 Documentation Library, any information that this book contained regarding AIX system security, or any security–related topic, has moved. For all security–related information, see the *AIX 5L Version 5.2 Security Guide*.

This edition supports the release of AIX 5L Version 5.2 with the 5200–01 Recommended Maintenance package. Any specific references to this maintenance package are indicated as *AIX 5.2 with 5200–01*.

Who Should Use This Book

This book is intended for system administrators who perform system management tasks that involve communication within a network.

Highlighting

The following highlighting conventions are used in this book:

Bold	Identifies commands, keywords, files, directories, and other items whose names are predefined by the system.
Italics	Identifies parameters whose actual names or values are to be supplied by the user.
`Monospace`	Identifies examples of specific data values, examples of text similar to what you might see displayed, examples of portions of program code similar to what you might write as a programmer, messages from the system, or information you should actually type.

Case–Sensitivity in AIX

Everything in the AIX operating system is case–sensitive, which means that it distinguishes between uppercase and lowercase letters. For example, you can use the **ls** command to list files. If you type LS, the system responds that the command is ”not found.” Likewise, **FILEA**, **FiLea**, and **filea** are three distinct file names, even if they reside in the same directory. To avoid causing undesirable actions to be performed, always ensure that you use the correct case.

ISO 9000

ISO 9000 registered quality systems were used in the development and manufacturing of this product.

Related Publications

The following book contains information about or related to communications:

- *AIX 5L Version 5.2 System Management Guide: Operating System and Devices*
- *AIX 5L Version 5.2 System Management Concepts: Operating System and Devices*
- *AIX 5L Version 5.2 System User's Guide: Communications and Networks*
- *AIX 5L Version 5.2 General Programming Concepts: Writing and Debugging Programs*
- *AIX 5L Version 5.2* Network Information Service *(NIS and NIS+) Guide*
- *AIX 5L Version 5.2 Commands Reference*
- *AIX 5L Version 5.2 Installation Guide and Reference*
- *AIX 5L Version 5.2 Security Guide*

Contents

Chapter 1. How To's for Network Administration Tasks

This chapter provides the following How–To instructions for performing common network administration tasks:

- Upgrade to IPv6 with IPv4 Configured on page 1-2
- Upgrade to IPv6 with IPv4 not configured in AIX 5.2 and later on page 1-4
- Migrate from SNMPv1 to SNMPv3 on page 1-7
- Create Users in SNMPv3 on page 1-11
- Dynamically update authentication and privacy keys in SNMPv3 on page 1-16
- Create a Local Alias for Mail on page 1-19
- Configure Domain Name Servers on page 1-20

Upgrade to IPv6 with IPv4 Configured

This scenario leads you through a manual upgrade from IPv4 to IPv6. The network used in this example consists of a router and two subnets. There are two hosts on each subnet: the router, and another host. You will upgrade each machine on this network to IPv6. By the end of the scenario, the router will advertise prefix `fec0:0:0:aaaa::/64` on network interface `en0` and prefix `fec0:0:0:bbbb::/64` on network interface `en1`. You will first configure the machines to temporarily support IPv6 so that you can test them. You will then configure the machines so they will be IPv6–ready at boot time.

If you are running AIX 5.2 and do not have your IPv4 settings configured, see Upgrade to IPv6 with IPv4 not configured in AIX 5.2 and later on page 1-4.

Step 1. Set up the hosts for IPv6

On the hosts on both subnets, do the following:

1. Make sure IPv4 is configured by typing the following command:

```
netstat –ni
 Your results should look similar to the following:

Name  Mtu   Network     Address            Ipkts Ierrs    Opkts Oerrs
Coll
 en0   1500  link#2      0.6.29.4.55.ec      279393     0     2510     0
   0
 en0   1500  9.3.230.64  9.3.230.117         279393     0     2510     0
   0
 lo0   16896 link#1                             913     0      919     0
   0
 lo0   16896 127         127.0.0.1              913     0      919     0
   0
 lo0   16896 ::1                                913     0      919     0
   0
```

2. With root authority, configure your IPv6 settings by typing the following command:

```
autoconf6
```

3. Rerun the following command:

```
   netstat –ni
    Your results should look similar to the following:

Name  Mtu   Network     Address            Ipkts Ierrs    Opkts Oerrs  Coll
 en0   1500  link#2      0.6.29.4.55.ec      279679     0     2658     0
 0
 en0   1500  9.3.230.64  9.3.230.117         279679     0     2658     0
 0
 en0   1500  fe80::206:29ff:fe04:55ec        279679     0     2658     0
 0
 sit0  1480  link#3      9.3.230.117               0    0        0     0
 0
 sit0  1480  ::9.3.230.117                         0    0        0     0
 0
 lo0   16896 link#1                            2343     0     2350     0
 0
 lo0   16896 127         127.0.0.1             2343     0     2350     0
 0
 lo0   16896 ::1                               2343     0     2350     0
 0
```

4. Start the **ndpd–host** daemon by typing the following command:

```
startsrc –s ndpd–host
```

The host is now IPv6–ready. Repeat this procedure for every host on each subnet.

Step 2. Set up the router for IPv6

1. Make sure that the IPv4 settings are configured by typing the following command:

   ```
   netstat –ni
   ```

2. With root authority, type the following command:

   ```
   autoconf6
   ```

3. Manually configure site–local addresses on the router's interfaces belonging to each of the two subnets by typing the following commands:

   ```
   # ifconfig en0 inet6 fec0:0:0:aaaa::/64 eui64 alias
    # ifconfig en1 inet6 fec0:0:0:bbbb::/64 eui64 alias
   ```

   You will need to do this for every subnet that your router is sending packets to.

4. To activate IPv6 forwarding, type the following:

   ```
   no –o ip6forwarding=1
   ```

5. To start the **ndpd–router** daemon, type the following:

   ```
   startsrc –s ndpd–router
   ```

   The **ndpd–router** daemon will advertise prefixes corresponding to the site–local addresses that you configured on the router. In this case, the ndpd–router will advertise prefix ec0:0:0:aaaa::/64 on en0 and prefix fec0:0:0:bbbb::/64 on en1.

Step 3. Set up IPv6 to be configured on the hosts at boot time

Your newly configured IPv6 will be deleted when you reboot the machine. To enable IPv6 host functionality every time you reboot, do the following:

1. Open the **/etc/rc.tcpip** file using your favorite text editor.

2. Uncomment the following lines in that file:

   ```
   # Start up autoconf6 process
    start /usr/sbin/autoconf6 ""

   # Start up ndpd–host daemon
    start /usr/sbin/ndpd–host "$src_running"
   ```

When you reboot, your IPv6 configuration will be set. Repeat this process for each host.

Step 4: Set up IPv6 to be configured on the router at boot time

Your newly configured IPv6 will be deleted when you reboot. To enable IPv6 router functionality every time you reboot, do the following:

1. Open the **/etc/rc.tcpip** file in your favorite text editor.

2. Uncomment the following line in that file:

   ```
   # Start up autoconf6 process
    start /usr/sbin/autoconf6 ""
   ```

3. Add the following lines immediately after the line that you just uncommented in the previous step:

   ```
   # Configure site–local addresses for router
    ifconfig en0 inet6 fec0:0:0:aaaa::/ eui64 alias
    ifconfig en1 inet6 fec0:0:0:bbbb::/ eui64 alias
   ```

   In this scenario, our network has only two subnets, en0 and en1. You will need to add a line to this file for every subnet that your router is sending packets to.

4. Uncomment the following line in the file:

   ```
   # Start up ndpd–router daemon
    start /usr/sbin/ndpd–router "$src_running"
   ```

When you reboot, IPv6 will be automatically started.

Upgrade to IPv6 with IPv4 not configured in AIX 5.2 and later

This scenario shows how to set up hosts and a router for IPv6 without IPv4 settings configured. The network used in this example consists of a router and two subnets. There are two hosts on each subnet: the router, and another host. By the end of the scenario, the router will advertise prefix `fec0:0:0:aaaa::/64` on network interface `en0` and prefix `fec0:0:0:bbbb::/64` on network interface `en1`. You will first configure the machines to temporarily support IPv6 so that you can test them. You will then configure the machines so they will be IPv6–ready at boot time.

This scenario assumes that the **bos.net.tcp.client** fileset is installed.

To upgrade to IPv6 with IPv4 already configured, see Upgrade to IPv6 with IPv4 Configured on page 1-2.

Step 1: Set up the hosts for IPv6

1. With root authority, type the following command on each host on the subnet:

   ```
   autoconf6 –A
    This will bring up all IPv6–capable interfaces on the system.
   ```

   Note:: To bring up a subset of interfaces, use the `–i` flag. For example, `autoconf6 –i en0 en1` will bring up interfaces `en0` and `en1`.

2. Type the following command to view your interfaces:

   ```
   netstat –ni
    Your results should look similar to the following:

   Name  Mtu   Network     Address           Ipkts Ierrs    Opkts Oerrs
   Coll
    en0   1500  link#3      0.4.ac.17.b4.11          7     0       17     0
      0
    en0   1500  fe80::204:acff:fe17:b411             7     0       17     0
      0
    lo0   16896 link#1                             436     0      481     0
      0
    lo0   16896 127         127.0.0.1              436     0      481     0
      0
    lo0   16896 ::1                                436     0      481     0
      0
   ```

3. Start the **ndpd–host** daemon by typing the following command:

   ```
   startsrc –s ndpd–host
   ```

Step 2: Set up the router for IPv6

1. With root authority, type the following command on the router host:

   ```
   autoconf6 –A
    This will bring up all IPv6–capable interfaces on the system.
   ```

   Note:: To bring up a subset of interfaces, use the `–i` flag. For example, `autoconf6 –i en0 en1` will bring up interfaces `en0` and `en1`.

   Your results should look similar to the following:

```
Name  Mtu   Network     Address            Ipkts Ierrs    Opkts Oerrs
Coll
 en1   1500  link#2      0.6.29.dc.15.45          0     0        7     0
  0
 en1   1500  fe80::206:29ff:fedc:1545             0     0        7     0
  0
 en0   1500  link#3      0.4.ac.17.b4.11          7     0       17     0
  0
 en0   1500  fe80::204:acff:fe17:b411             7     0       17     0
  0
 lo0   16896 link#1                             436     0      481     0
  0
 lo0   16896 127         127.0.0.1              436     0      481     0
  0
 lo0   16896 ::1                                436     0      481     0
  0
```

2. Manually configure site–local addresses on the router's interfaces belonging to each of the two subnets by typing the following commands:

   ```
   # ifconfig en0 inet6 fec0:0:0:aaaa::/64 eui64 alias
    # ifconfig en1 inet6 fec0:0:0:bbbb::/64 eui64 alias
   ```

   Note:: You will need to do this for every subnet that your router is sending packets to.

3. To activate IPv6 forwarding, type the following:

   ```
   no –o ip6forwarding=1
   ```

4. To start the **ndpd–router** daemon, type the following:

   ```
   startsrc –s ndpd–router
   ```

   The **ndpd–router** daemon will advertise prefixes corresponding to the site–local addresses that you configured on the router. In this case, the ndpd–router will advertise prefix ec0:0:0:aaaa::/64 on en0 and prefix fec0:0:0:bbbb::/64 on en1.

Step 3. Set up IPv6 to be configured on the hosts at boot time

After completing Step 1 for each host, IPv6 will be deleted when you reboot the machine. To enable IPv6 host functionality every time you reboot, do the following:

1. Open the **/etc/rc.tcpip** file using your favorite text editor.
2. Uncomment the following lines in that file:

   ```
   # Start up autoconf6 process
    start /usr/sbin/autoconf6 ""

   # Start up ndpd–host daemon
    start /usr/sbin/ndpd–host "$src_running"
   ```

3. Add the `–A` flag to `start /usr/sbin/autoconf6 ""`:

   ```
   start /usr/sbin/autoconf6 "" –A
   ```

4. Repeat this process for each host.

When you reboot, IPv6 will be automatically started.

Step 4: Set up IPv6 to be configured on the router at boot time

After completing Step 2 for your router, IPv6 will be deleted when you reboot. To enable IPv6 router functionality every time you reboot, do the following:

1. Open the **/etc/rc.tcpip** file in your favorite text editor.
2. Uncomment the following line in that file:

   ```
   # Start up autoconf6 process
    start /usr/sbin/autoconf6 ""
   ```

3. Add the `–A` flag to that line:

   ```
   start /usr/sbin/autoconf6 ”” –A
   ```

4. Add the following lines immediately after the line that you just uncommented in the previous step:

   ```
   # Configure site–local addresses for router
    ifconfig en0 inet6 fec0:0:0:aaaa::/ eui64 alias
    ifconfig en1 inet6 fec0:0:0:bbbb::/ eui64 alias
   ```

   In this scenario, our network has only two subnets, en0 and en1. You will need to add a line to this file for every subnet that your router is sending packets to.

5. Uncomment the following line in the file:

   ```
   # Start up ndpd–router daemon
    start /usr/sbin/ndpd–router ”$src_running”
   ```

6. Run the following command to enable IP forwarding at boot time:

   ```
   no –r –o ip6forwarding=1
   ```

When you reboot, IPv6 will be automatically started.

Migrate from SNMPv1 to SNMPv3

This scenario shows a typical migration from SNMPv1 to SNMPv3.

In AIX 5.2, the default SNMP agent running at system boot time is the non–encrypted version of SNMPv3. SNMPv3 uses the **/etc/snmpdv3.conf** file as its configuration file. Any parameters that you had configured in the **/etc/snmpd.conf** file, which is used by SNMPv1 in AIX 5.1 and earlier, will need to be manually migrated to the **/etc/snmpdv3.conf** file.

In this scenario, the communities and traps that were configured in the **/etc/snmpd.conf** file will be migrated to the **/etc/snmpdv3.conf** file. By the end of the scenario, SNMPv3 will provide identical functionality that SNMPv1 offered. If you did not configure any of your own SNMPv1 communities or traps, there is no need for you to complete this procedure.

This file does not contain any information about features available in SNMPv3. For information about creating users using SNMPv3 features not available in SNMPv1, see Create Users in SNMPv3 on page 1-11.

The following file is the example **/etc/snmpd.conf** file that is going to be migrated. The following communities are configured: `daniel`, `vasu`, and `david`. These communities must be migrated manually.

```
logging         file=/usr/tmp/snmpd.log         enabled
 logging         size=0                          level=0

 community       daniel       0.0.0.0     0.0.0.0         readWrite
1.17.35
 community       vasu         9.3.149.49  255.255.255.255 readOnly   10.3.5
 community       david        9.53.150.67 255.255.255.255 readWrite
1.17.35

 view 1.17.35    udp  icmp  snmp 1.3.6.1.2.1.25
 view 10.3.5     system interfaces tcp icmp

 trap            daniel          9.3.149.49      1.17.35   fe
 trap            vasu            9.3.149.49      10.3.5   fe
 trap            david           9.53.150.67     1.17.35   fe

 smux            1.3.6.1.4.1.2.3.1.2.3.1.1         sampled_password  #
sampled
```

To complete the steps in this scenario, refer to your **/etc/snmpd.conf** file. Have a copy of that file ready when you start this procedure.

Step 1. Migrate the community information

The community names in the **/etc/snmpd.conf** file become part of the `VACM_GROUP` entries in the **/etc/snmpdv3.conf** file. Each community must be placed in a group. You will then give the groups the view and access permissions needed.

1. With root authority, open the **/etc/snmpdv3.conf** file with your favorite text editor. Locate the `VACM_GROUP` entries in the file.
2. Create a `VACM_GROUP` entry for each community that you want to migrate. If multiple communities are going to share the same view and access permissions, you need to create only one group for them. The community names in the **/etc/snmpd.conf** file become the *`securityName`* values for the `VACM_GROUP` entries. In this scenario, the following entries were added for `vasu`, `daniel`, and `david`:

```
#--------------------------------------------------------------------
# VACM_GROUP entries
#    Defines a security group (made up of users or communities)
#    for the View–based Access Control Model (VACM).
# Format is:
#  groupName securityModel securityName storageType
VACM_GROUP group2 SNMPv1  vasu –
VACM_GROUP group3 SNMPv1  daniel –
VACM_GROUP group3 SNMPv1  david –
#--------------------------------------------------------------------
```

- *`groupName`* can be any value you choose, except `group1`.
- *`securityModel`* remains `SNMPv1` because we are migrating the SNMPv1 communities.
- In this scenario, `daniel` and `david` share the same view and access permissions in the **/etc/snmpd.conf** file. Therefore, they are both members of `group3` in the **/etc/snmpdv3.conf** file. Community `vasu` is placed in a different group because its view and access permissions are different than those for `david` and `daniel`.

The communities are now placed in groups.

Step 2. Migrate the view information

The view information in the **/etc/snmpd.conf** file will become `COMMUNITY`, `VACM_VIEW`, and `VACM_ACCESS` entries in the **/etc/snmpdv3.conf** file. These entries will determine the view and access permissions for each group.

1. Create `COMMUNITY` entries for `daniel`, `vasu`, and `david`, maintaining the same IP addresses for `netAddr` and `netMask` as were specified in the **/etc/snmpd.conf** file.

```
#------------------------------------------------------------------------
# COMMUNITY
#    Defines a community for community–based security.
# Format is:
#  communityName securityName securityLevel netAddr netMask storageType
COMMUNITY public    public     noAuthNoPriv 0.0.0.0     0.0.0.0         –
COMMUNITY daniel    daniel     noAuthNoPriv 0.0.0.0     0.0.0.0         –
COMMUNITY vasu      vasu       noAuthNoPriv 9.3.149.49  255.255.255.255 –
COMMUNITY david     david      noAuthNoPriv 9.53.150.67 255.255.255.255 –
#------------------------------------------------------------------------
```

2. Create a `VACM_VIEW` entry for every MIB object or variable that each group has access to. According to the **/etc/snmpd.conf** file, `daniel` and `david` have access to `udp`, `icmp`, `snmp`, and `1.3.6.1.2.1.25` (host subtree as defined in RFC 1514), and `vasu` has access to `system`, `interfaces`, `tcp`, and `icmp`. These view entries are migrated to the **/etc/snmpdv3.conf** file as follows:

```
#--------------------------------------------------------------------
# VACM_VIEW entries
#    Defines a particular set of MIB data, called a view, for the
#    View–based Access Control Model.
# Format is:
#  viewName viewSubtree viewMask viewType storageType

VACM_VIEW group2View        system                      – included –
VACM_VIEW group2View        interfaces                  – included –
VACM_VIEW group2View        tcp                         – included –
VACM_VIEW group2View        icmp                        – included –

VACM_VIEW group3View        udp                         – included –
VACM_VIEW group3View        icmp                        – included –
VACM_VIEW group3View        snmp                        – included –
VACM_VIEW group3View        1.3.6.1.2.1.25              – included –
#---------------------------------------------------------------------
```

3. Define access permissions to the MIB variables defined in the `VACM_VIEW` entries by adding `VACM_ACCESS` entries. In the **/etc/snmpd.conf** file, `daniel` and `david` both have `readWrite` permission to their MIB variables, whereas `vasu` has `readOnly`.

Define these permissions by adding `VACM_ACCESS` entries. In this scenario, we gave `group2` (`vasu`) the `group2View` for `readView`, but gave it – for `writeView` because `vasu` had `readOnly` in the **/etc/snmpd.conf** file. We gave `group3` (`daniel` and `david`) the `group3View` for both `readView` and `writeView` because those groups had `readWrite` access in **/etc/snmpd.conf**. See the following example.

```
#------------------------------------------------------------------------
# VACM_ACCESS entries
#    Identifies the access permitted to different security groups
#    for the View–based Access Control Model.
# Format is:
# groupName contextPrefix contextMatch securityLevel securityModel
readView writeView notifyView storageType
VACM_ACCESS  group1 – – noAuthNoPriv SNMPv1  defaultView – defaultView –
VACM_ACCESS  group2 – – noAuthNoPriv SNMPv1  group2View – group2View –
VACM_ACCESS  group3 – – noAuthNoPriv SNMPv1  group3View group3View
group3View –
#------------------------------------------------------------------------
```

Step 3. Migrate the trap information

The trap entries in the **/etc/snmpd.conf** file will become the `NOTIFY`, `TARGET_ADDRESS`, and `TARGET_PARAMETERS` entries in the **/etc/snmpdv3.conf** file. However, only the `TARGET_ADDRESS` and `TARGET_PARAMETERS` will need to be migrated.

1. The IP addresses listed in the trap entries in the **/etc/snmpd.conf** file become part of the `TARGET_ADDRESS` entries in the **/etc/snmpdv3.conf** file. This line specifies the host where the trap will be sent. You can define the `targetParams` entries. In this scenario, we use `trapparms1`, `trapparms2`, `trapparms3`, and `trapparms4`, which will be defined in the `TARGET_PARAMETERS` entries.

```
#------------------------------------------------------------------------
# TARGET_ADDRESS
#    Defines a management application's address and parameters
#    to be used in sending  notifications.
# Format is:
#  targetAddrName tDomain tAddress tagList targetParams timeout
retryCount storageType
TARGET_ADDRESS Target1 UDP 127.0.0.1       traptag trapparms1 – – –
TARGET_ADDRESS Target2 UDP 9.3.149.49   traptag trapparms2 – – –
TARGET_ADDRESS Target3 UDP 9.3.149.49   traptag trapparms3 – – –
TARGET_ADDRESS Target4 UDP 9.53.150.67  traptag trapparms4 – – –
#------------------------------------------------------------------------
```

2. The community names specified in the trap entries in the **/etc/snmpd.conf** file become part of the `TARGET_PARAMETERS` entries in the **/etc/snmpdv3.conf** file. The community names must be mapped to a specific `TARGET_ADDRESS` entry using the `targetParams` values. For example, community `daniel` is mapped with `trapparms2`, which, under the `TARGET_ADDRESS` entry, maps to IP address `9.3.149.49`. Community `daniel` and IP address `9.3.149.49` were originally a `trap` entry in the **/etc/snmpd.conf** file. See the following example:

```
#------------------------------------------------------------------------
# TARGET_PARAMETERS
#    Defines the message processing and security parameters
#    to be used in sending notifications to a particular management
target.
# Format is:
#  paramsName mpModel securityModel securityName securityLevel
storageType
TARGET_PARAMETERS trapparms1 SNMPv1  SNMPv1  public  noAuthNoPriv –
TARGET_PARAMETERS trapparms2 SNMPv1  SNMPv1  daniel  noAuthNoPriv –
TARGET_PARAMETERS trapparms3 SNMPv1  SNMPv1  vasu    noAuthNoPriv –
TARGET_PARAMETERS trapparms4 SNMPv1  SNMPv1  david   noAuthNoPriv –
#-----------------------------------------------------------------------
```

3. The `trapmask` information in the **/etc/snmpd.conf** file does not migrate to the **/etc/snmpdv3.conf** file.

Step 4. Migrate the smux information

If you have smux information that you need to migrate, you can copy those lines directly into the new file. In this scenario, the `sampled` smux entry was configured in the **/etc/snmpd.conf** file. That line must be copied to the **/etc/snmpdv3.conf** file.

```
#-------------------------------------------------------------------------
#       smux <client OIdentifier> <password> <address> <netmask>
smux            1.3.6.1.4.1.2.3.1.2.3.1.1         sampled_password  #
sampled
#-------------------------------------------------------------------------
```

Step 5. Stop and Start the snmpd daemon

After the migration of the **/etc/snmpd.conf** file to the **/etc/snmpdv3.conf** file is complete, stop and then start the **snmpd** daemon. You will need to stop and start the **snmpd** daemon each time you make changes to the **/etc/snmpdv3.conf** file.

1. Type the following command to stop the daemon:

   ```
   stopsrc –s snmpd
   ```

2. Type the following command to restart the daemon:

   ```
   startsrc –s snmpd
   ```

Note:: Simply refreshing the SNMPv3 agent will not work as it did in SNMPv1. If you make changes to the **/etc/snmpdv3.conf** file, you must stop and start the daemon as instructed above. The dynamic configuration function supported in SNMPv3 will not allow you to refresh.

Create Users in SNMPv3

This scenario shows how to create a user in SNMPv3 by manually editing the **/etc/snmpdv3.conf** and **/etc/clsnmp.conf** files.

User `u1` will be created in this scenario. User `u1` will be given authorization keys, but will not be given privacy keys (which are available only if you have the **snmp.crypto** fileset installed). The HMAC–MD5 protocol will be used to create u1's authorization keys. After u1 is configured, it will be put into a group, after which that group will have its view and access permissions defined. Finally, trap entries for u1 will be created.

Each individual value used in the **/etc/snmpdv3.conf** and **/etc/clsnmp.conf** files must not exceed 32 bytes.

Step 1. Create the user

1. Decide which security protocols you want to use, either HMAC–MD5 or HMAC–SHA. In this scenario, HMAC–MD5 will be used.
2. Generate the authentication keys by using the **pwtokey** command. Your output may look different based on the authentication protocol you are using and if you are using privacy keys. These keys will be used in the **/etc/snmpdv3.conf** and **/etc/clsnmp.conf** files. The command used for user `u1` follows:

   ```
   pwtokey –p HMAC–MD5 –u auth   anypassword   9.3.230.119
   ```

   The IP address specified is the IP address where the agent is running. The password can by any password, but be sure to save it in a secure place for future use. The output should look similar to the following:

   ```
   Display of 16 byte HMAC–MD5 authKey:
      63960c12520dc8829d27f7fbaf5a0470

    Display of 16 byte HMAC–MD5 localized authKey:
      b3b6c6306d67e9c6f8e7e664a47ef9a0
   ```

3. With root authority, open the **/etc/snmpdv3.conf** file with your favorite text editor.
4. Create a user by adding a `USM_USER` entry following the format given in the file. The *`authKey`* value will be the localized authentication key that was generated using the **pwtokey** command. The entry for user `u1` follows:

   ```
   #-----------------------------------------------------------------------
   # USM_USER entries
   #    Defines a user for the User–based Security Model (USM).
   # Format is:
   #  userName engineID authProto authKey privProto privKey keyType
   storageType
   #
   USM_USER u1 – HMAC–MD5 b3b6c6306d67e9c6f8e7e664a47ef9a0 – – L –
   #-----------------------------------------------------------------------
   ```

   - *`userName`* is the name of the user. In this case, it is `u1`.
   - *`authProto`* must be the protocol that you used when you created the keys. In this case, it is `HMAC–MD5`.
   - *`authKey`* is the localized authentication key that was created using the **pwtokey** command.
   - *`privProto`* and *`privkey`* are not specified because we are not using the privacy keys in this scenario.
   - *`keyType`* is `L` because we are using the localized authentication key.
5. Save and close the **/etc/snmpdv3.conf** file.
6. Open the **/etc/clsnmp.conf** file on the SNMP manager with your favorite text editor.

7. Add the new user according to the format given in the file. The entry for `u1` follows:

   ```
   #------------------------------------------------------------------------
   #
   # Format of entries:
   # winSnmpName targetAgent admin secName password context secLevel
   authProto authKey privProto privKey
   #
   user1  9.3.230.119  SNMPv3  u1  –  –  AuthNoPriv  HMAC–MD5
   63960c12520dc8829d27f7fbaf5a0470  –  –
   #------------------------------------------------------------------------
   ```

   - *`winSnmpName`* can be any value. This value will be used when making SNMP requests using the **clsnmp** command.
   - *`targetAgent`* is the IP address where the agent is running, which was also used in creating the authentication keys.
   - *`admin`* is set to `SNMPv3` because we will be sending SNMPv3 requests.
   - *`secName`* is the name of the user that you are creating. In this case, it is `u1`.
   - *`seclevel`* is set to `AuthNoPriv` because it is being configured to use authentication but not privacy (as a result, there are no values for *`privProto`* and *`privKey`*).
   - *`authproto`* is set to the authentication protocol that was used in creating the authentication keys.
   - *`authKey`* is the non–localized key that was generated by the **pwtokey** command.

8. Save and close the **/etc/clsnmp.conf** file.

Step 2. Configure the group

The user must now be placed in a group. If you already have a group that is configured with all of the view and access permissions that you want to give this user, you can put this user in that group. If you want to give this user view and access permissions that no other groups have, or if you do not have any groups configured, create a group and add this user to it.

To add the user to a new group, create a new `VACM_GROUP` entry in the **/etc/snmpdv3.conf** file. The group entry for `u1` follows:

```
#-------------------------------------------------------------
# VACM_GROUP entries
#    Defines a security group (made up of users or communities)
#    for the View–based Access Control Model (VACM).
# Format is:
#  groupName securityModel securityName storageType
VACM_GROUP group1 USM u1 –
#-------------------------------------------------------------
```

- *`groupName`* can be any name. It becomes that name of your group. In this case, it is `group1`.
- *`securityModel`* is set to `USM`, which takes advantage of the SNMPv3 security features.
- *`securityName`* is the name of the user. In this case, it is `u1`.

Step 3. Configure view and access permissions

The view and access permissions must be set for the new group that was just created. These permissions are set by adding `VACM_VIEW` and `VACM_ACCESS` entries to the **/etc/snmpdv3.conf** file.

1. Decide what view and access permissions you want the new group to have.
2. Add `VACM_VIEW` entries to the **/etc/snmpdv3.conf** file to define what MIB objects the group can access. In this scenario, `group1` will have access to the `interfaces`, `tcp`,

icmp, and system MIB subtrees. However, we will restrict group1 's access to the sysObjectID MIB variable within the system MIB subtree.

```
#---------------------------------------------------------------
# VACM_VIEW entries
#    Defines a particular set of MIB data, called a view, for the
#    View–based Access Control Model.
# Format is:
#  viewName viewSubtree viewMask viewType storageType
VACM_VIEW group1View        interfaces     – included –
VACM_VIEW group1View        tcp           – included –
VACM_VIEW group1View        icmp              – included –
VACM_VIEW group1View        system         – included –
VACM_VIEW group1View        sysObjectID    – excluded –
#---------------------------------------------------------------
```

- *viewName* is the name of the view. In this scenario, it is group1View.
- *viewSubtree* is the MIB subtree that you want to give access to.
- *viewType* determines whether the MIB subtrees defined are included in the view. In this case, all subtrees are included, but the MIB variable sysObjectID, which is part of the system subtree, is excluded.

3. Add a VACM_ACCESS entry to the **/etc/snmpdv3.conf** file to define the permissions that the group has to the MIB objects specified above. For group1, read only access is given.

```
#-----------------------------------------------------------------------
# VACM_ACCESS entries
#    Identifies the access permitted to different security groups
#    for the View–based Access Control Model.
# Format is:
# groupName contextPrefix contextMatch securityLevel securityModel
readView writeView notifyView storageType
VACM_ACCESS  group1 – – AuthNoPriv USM group1View – group1View –
#----------------------------------------------------------------------
```

- *groupName* is the name of the group. In this case, it is group1.
- *securityLevel* is the level of security that is being used. In this scenario, authentication keys are used but not privacy keys. The value is therefore set to AuthNoPriv.
- *securityModel* is the security model that you are using (SNMPv1, SNMPv2c, or USM). In this scenario, it is set to USM to allow the SNMPv3 security features to be used.
- *readView* determines which VACM_VIEWs the group has read access to. In this scenario, group1View is given, which gives group1 read access to the group1View VACM_VIEW entries.
- *writeView* determines which VACM_VIEWs the group has write access to. In this scenario, no write access is given to group1.
- *notifyView* specifies the name of the view to be applied when a trap is performed under control of the entry in the access table.

Note:: In some cases, multiple VACM_ACCESS entries for one group may be necessary. If users in the group have different authentication and privacy settings (noAuthNoPriv, AuthNoPriv, or AuthPriv) multiple VACM_ACCESS entries are required with the securityLevel parameter set accordingly.

Step 4. Configure trap entries for the user

Trap entries in SNMPv3 are created by adding NOTIFY, TARGET_ADDRESS and TARGET_PARAMETERS entries to the **/etc/snmpdv3.conf** file. The TARGET_ADDRESS entry will specify where you want the traps to be sent, and the TARGET_PARAMETERS entry will map the TARGET_ADDRESS information to group1.

The NOTIFY entry has been configured by default. Following is the default NOTIFY entry:

```
NOTIFY notify1 traptag trap –
```

In this scenario, we use the value that is specified in the default entry, traptag.

1. Add a TARGET_ADDRESS entry to specify where you want traps to be sent.

```
#-------------------------------------------------------------------------
# TARGET_ADDRESS
#    Defines a management application’s address and parameters
#    to be used in sending  notifications.
# Format is:
#  targetAddrName tDomain tAddress tagList targetParams timeout
retryCount storageType
#-------------------------------------------------------------------------
TARGET_ADDRESS Target1 UDP 9.3.207.107     traptag trapparms1 – – –
```

   - *targetAddrName* can be any name. In this scenario, we used Target1.
   - *tAddress* is the IP address where the traps for the group should be sent.
   - *tagList* is the name configured in the NOTIFY entry. In this scenario, it is traptag.
   - *targetParams* can be any value. We used is trapparms1, which will be used in the TARGET_PARAMETERS entry.

2. Add a TARGET_PARAMETERS entry.

```
#------------------------------------------------------------------------
# TARGET_PARAMETERS
#    Defines the message processing and security parameters
#    to be used in sending notifications to a particular management
target.
# Format is:
#  paramsName mpModel securityModel securityName securityLevel
storageType
#-------------------------------------------------------------------------
TARGET_PARAMETERS trapparms1 SNMPv3  USM     u1       AuthNoPriv
```

   - *paramsName* is the same as the targetParams value in the TARGET_ADDRESS entry, which, in this case, is trapparms1.
   - *mpModel* is the version of SNMP being used.
   - *securityModel* is the security model that you are using (SNMPv1, SNMPv3, or USM). In this scenario, it is set to USM to allow the SNMPv3 security features to be used.
   - *securityName* is the user name specified in the USM_USER entry, which, in this case, is u1.
   - *securityLevel* is set to AuthNoPriv because we are using authentication keys but not privacy keys.

Step 5. Stop and start the snmpd daemon

After making the changes the **/etc/snmpdv3.conf** file, stop and the start the **snmpd** daemon.

1. Type the following command to stop the **snmpd** daemon:

   ```
   stopsrc –s snmpd
   ```

2. Type the following command to start the **snmpd** daemon:

   ```
   startsrc –s snmpd
   ```

The new settings will now take effect.

Note:: Simply refreshing the SNMPv3 agent using **refresh –s snmpd** will not work as it did in SNMPv1. If you make changes to the **/etc/snmpdv3.conf** file, you must stop and start the daemon as instructed above. The dynamic configuration function supported in SNMPv3 will not allow you to refresh.

Step 6. Test your configuration

To verify that your configuration is correct, you can run the following command on the SNMP manager .

```
clsnmp –h user1 walk   mib
```

where *mib* is a MIB subtree to which the user has access. In this scenario, it could be `interfaces`, `tcp`, `icmp`, or `system`. If the configuration is correct, you will see the information from the specified subtree.

If you did not get the correct output, review the steps in this document and verify that you have entered all information correctly.

Dynamically update authentication and privacy keys in SNMPv3

This scenario shows how to dynamically update the authentication keys for a user in SNMPv3. In this scenario, user `u4` will update the authentication keys for user `u8`. Both users `u4` and `u8` have already had authentication keys created based on password `defaultpassword` and IP address `9.3.149.49`, and everything is working.

During this scenario, new keys will be created for user `u8` and the **/etc/snmpdv3.conf** file will be dynamically updated. The authentication key for user `u8` in the manager side's **/etc/clsnmp.conf** file will then need to be manually edited to reflect the new keys.

Make a backup of the **/etc/snmpdv3.conf** file on the SNMP agent and a backup of the **/etc/clsnmp.conf** file on the SNMP manager before you start this procedure.

Below is the **/etc/snmpdv3.conf** file that will be dynamically updated:

```
USM_USER u4 – HMAC–MD5 18a2c7b78f3df552367383eef9db2e9f – – N –
 USM_USER u8 – HMAC–SHA 754ebf6ab740556be9f0930b2a2256ca40e76ef9 – – N –

 VACM_GROUP group1 SNMPv1  public  –
 VACM_GROUP group2 USM u4 –
 VACM_GROUP group2 USM u8 –

 VACM_VIEW defaultView        internet                   – included –

 VACM_ACCESS  group1 – – noAuthNoPriv SNMPv1  defaultView – defaultView –
 VACM_ACCESS  group2 – – noAuthNoPriv USM defaultView defaultView
defaultView –
 VACM_ACCESS  group2 – – AuthNoPriv USM defaultView defaultView defaultView
–
 VACM_ACCESS  group2 – – AuthPriv USM defaultView defaultView defaultView –

 NOTIFY notify1 traptag trap –

 TARGET_ADDRESS Target1 UDP 127.0.0.1       traptag trapparms1 – – –
 TARGET_ADDRESS Target2 UDP 9.3.149.49       traptag trapparms2 – – –
 TARGET_ADDRESS Target3 UDP 9.3.149.49       traptag trapparms3 – – –
 TARGET_ADDRESS Target4 UDP 9.3.149.49       traptag trapparms4 – – –

 TARGET_PARAMETERS trapparms1 SNMPv1  SNMPv1  public  noAuthNoPriv –
 TARGET_PARAMETERS trapparms3 SNMPv2c  SNMPv2c  publicv2c  noAuthNoPriv –
 TARGET_PARAMETERS trapparms4 SNMPv3  USM     u4 AuthNoPriv –
```

Below is the **/etc/clsnmp.conf** file that will be updated for user `u8`:

```
testu4 9.3.149.49 snmpv3 u4 – – AuthNoPriv HMAC–MD5
18a2c7b78f3df552367383eef9db2e9f – –
 testu8 9.3.149.49 snmpv3 u8 – – AuthNoPriv HMAC–SHA
754ebf6ab740556be9f0930b2a2256ca40e76ef9 – –
```

Follow these steps to update your password and authentication keys.

1. On the SNMP manager side, run the **pwchange** command. In this scenario, we ran the following command:

   ```
   pwchange –u auth –p HMAC–SHA defaultpassword newpassword 9.3.149.49
   ```

   This command will generate a new authentication key.

   – *`–u auth`* specifies that only an authentication key will be created. If you are updating privacy keys as well, use `–u all`.

   – *`–p HMAC–SHA`* specifies the protocol that will be used to create the authentication key. If you are updating privacy keys as well, us `–p all`.

- *defaultpassword* is the password used to create the latest authentication key (for example, if `bluepen` would have been used to create the latest authentication key, `bluepen` would be used here as well)
- *newpassword* is the new password that will be used to generate the authentication key. Keep this password for future reference
- *9.3.149.49* is the IP address where the SNMP agent is running.

This command produced the following output:

```
Dump of 40 byte HMAC–SHA authKey keyChange value:
   8173701d7c00913af002a3379d4b150a
   f9566f56a4dbde21dd778bb166a86249
   4aa3a477e3b96e7d
```

You will use this authentication key in the next step.

Note:: Keep the new passwords you use in a safe place. You will need to use them again when making changes in the future.

2. On the SNMP manager, user `u4` will change the authentication key for user `u8` by entering the following command:

```
clsnmp –h testu4 set
usmUserAuthKeyChange.12.0.0.0.2.0.0.0.0.9.3.149.49.2.117.56

\'8173701d7c00913af002a3379d4b150af9566f56a4dbde21dd778bb166a862494aa3a47
7e3b96e7d\'h
```

- `testu4` is used because it is mapped to user `u4` in the **/etc/clsnmp.conf** file.
- The instance ID of *usmUserAuthKeyChange* includes, in decimal values, the engine ID of the SNMP agent where the update is taking place and username whose authentication key is being updated. The engine ID can be found in the **/etc/snmpd.boots** file (the **/etc/snmpd.boots** file contains two strings of numbers. The engine ID is the first string. Ignore the second string of numbers).

  The engine ID will need to be converted from hexadecimal values to decimal values in order to be used here. Each two numbers in the hexadecimal engine ID convert to one decimal value. For example, engine ID `000000020000000009039531` would be read as `00 00 00 02 00 00 00 00 09 03 95 31`. Each of those numbers must be converted to decimal values, resulting in, `0.0.0.2.0.0.0.0.9.3.149.49` (For a conversion table, see Appendix C. Conversion Table on page C-1 .). The first number in the string is the number of bytes in the decimal string. In this case, it is `12`, resulting in `12.0.0.0.2.0.0.0.0.9.3.149.49`.

  The following number is the number of bytes in the username, followed by the decimal values for the username itself. In this case, the username is `u8`. When converted to decimal values, `u8` becomes `117.56`. Because the username is 2 bytes long, the value representing the username becomes `2.117.56`. Add that to the end of the decimal engine ID (For a conversion table, see Appendix C. Conversion Table on page C-1 .).

  In this case, the result is `12.0.0.0.2.0.0.0.0.9.3.149.49.2.117.56`.
- The next value in the command is the new authentication key generated using the **pwchange** command in the previous step.

  Note:: If the user also has privacy keys configured, this procedure must be repeated to update the privacy keys. When updating the privacy keys, use the `usmUserPrivKeyChange` value instead of the `usmUserAuthKeyChange` value.

  Using `usmUserOwnAuthKeyChange` instead of `usmUserAuthKeyChange` will allow a user to change his or her own authentication key. For example, user `u4` could change its own authentication key using `usmUserOwnAuthKeyChange`.

The output of the command follows:

```
1.3.6.1.6.3.15.1.2.2.1.6.12.0.0.0.2.0.0.0.0.9.3.149.49.2.117.56 =
’8173701d7c00913af002a3379
 d4b150af9566f56a4dbde21dd778bb166a862494aa3a477e3b96e7d’h
```

After this command is completed, the **/etc/snmpdv3.conf** file will be automatically updated after five minutes on the SNMP agent side. You can also stop and start the SNMP daemon to update the file. The following entry for user u8 will be dynamically updated in the **/etc/snmpdv3.conf** file:

```
USM_USER u8 000000020000000009039531 HMAC–SHA
4be657b3ae92beee322ee5eaeef665b338caf2d9
 None – L nonVolatile
```

3. On the SNMP manager side, run the **pwtokey** command to generate the new authentication key based on the new password to place in the **/etc/clsnmp.conf** file. In this scenario, we ran the following command:

```
pwtokey –u auth –p HMAC–SHA  newpassword 9.3.149.49
```

   - *–u auth* specifies that only an authentication key will be created. If you are updating privacy keys as well, use `–u all`.
   - *–p HMAC–SHA* specifies the protocol that will be used in creating the authentication key. If you are updating privacy keys as well, use `–p all`.
   - The password used (in this case *newpassword*) must be the same as the password used when generating new authentication keys with the **pwchange** command.
   - The IP address used (in this case *9.3.149.49*) must be the IP address where the agent is running.

   The result gives the localized and non–localized authentication keys:

```
Display of 20 byte HMAC–SHA authKey:
   79ce23370c820332a7f2c7840c3439d12826c10d

 Display of 20 byte HMAC–SHA localized authKey:
   b07086b278163a4b873aace53a1a9ca250913f91
```

4. Open the **/etc/clsnmp.conf** file with your favorite text editor and place the non–localized authentication key in the line for the user whose keys are being updated. In this scenario, the entry is as follows:

```
testu8 9.3.149.49 snmpv3 u8 – – AuthNoPriv HMAC–SHA
79ce23370c820332a7f2c7840c3439d12826c10d – –
 Save and close the file.
```

5. Test the updated configuration by running the following command:

```
clsnmp –v –h testu8 walk   mib
```

   where *mib* is a MIB variable to which user u8 has read access. In this case, user u8 has access to internet.

Create a Local Alias for Mail

Creating local mail aliases allows you to create groups or distribution lists to which mail can be sent.

In this scenario, geo@medussa, mark@zeus, ctw@athena, and dsf@plato will be added to the `testers` mail alias. After the `testers` alias is created, glenda@hera will be given ownership of the alias.

After the `testers` alias has been added to the **/etc/mail/aliases** file, the aliases database will be recompiled using the **sendmail** command. After the database is recompiled, e–mail can be sent to the `testers` alias.

1. Open the **/etc/mail/aliases** file using your favorite text editor.
2. On a blank line, add the alias name, followed by a colon and a list of comma–separated recipients. For example, the following entry defines the `testers` alias:

   ```
   testers: geo@medussa, mark@zeus, ctw@athena, dsf@plato
   ```
3. Create an owner for the alias. If the **sendmail** command is unsuccessful in sending mail to the alias, it sends an error message to the owner.

   Add a line in the **/etc/mail/aliases** to specify the owner. The format for this line is `owner–` *`groupname`*`:` *`owner`*, where *`groupname`* is the name of the alias and *`owner`* is the e–mail address of the owner. In this example, `glenda@hera` is made the owner of the `testers` alias:

   ```
   testers: geo@medussa, mark@zeus, ctw@athena, dsf@plato
    owner–testers: glenda@hera
   ```
4. After the alias is created, run the `sendmail –bi` command to recompile the aliases database. You will need to run this command each time you update your **/etc/mail/aliases** file.

You can now send e–mail to the `testers` alias.

Configure Domain Name Servers

In this scenario, a master name server, slave name server, and hint name server will be configured to perform name resolution. Each name server will be a separate machine, and each will have an **/etc/named.conf** file configured, although the information in each will be different. The **/etc/named.conf** is read each time the **named** daemon is started, and it specifies what type of server it is (master, slave, or hint) and where it will get its name resolution data. Each of these name servers will be running BIND 8.

The master name server will be configured to provide name resolution for the `abc.aus.century.com` zone. In this scenario, the IP address of the master name server is `192.9.201.1`, and its host name is `venus.abc.aus.century.com`. It will provide name resolution for the venus, earth, mars, and jupiter host names. The **/etc/named.conf** file will be configured to specify that the **named** daemon should search the **/usr/local/domain** directory for its data files. The data files that will be configured for the master name server are **named.ca**, **named.abc.local**, **named.abc.data**, and **named.abc.rev**.

A slave name server will then be configured. The host name of the slave name server will be `earth.abc.aus.century.com`, and its IP address will be `192.9.201.5`. In the slave name server's **/etc/named.conf** file, we will specify the master name server's address so that the slave name server can replicate the master name server's **named.abc.data** and **named.abc.rev** files. In addition, the **named.ca** and **named.abc.local** data files will be configured for this server.

A hint name server will then be configured. The hint name server will store a local cache of host name and address mappings. If a requested address or host name is not in its cache, the hint server will contact the master name server, get the resolution information, and add it to its cache. In addition, the **named.ca** and **named.abc.local** data files will be configured for this server.

All information in the **named** data files (not the **/etc/named.conf** file) on the name servers must be in the Standard Resource Record Format. For explanations about the information about the **named** data files, see Standard Resource Record Format for TCP/IP in *AIX 5L Version 5.2 Files Reference*

The administrator for each of the name servers will be `gail.zeus.abc.aus.century.com`. This is specified in the local data files on each name server. In addition, in this scenario, the root name server is `relay.century.com` with IP address `129.114.1.2`.

At the end of this scenario, name resolution will be provided for the hosts venus, earth, mars, and jupiter. In addition, reverse name resolution (IP address–to–host name) will also be provided. When a request is received that cannot be resolved, the master name server will contact `relay.century.com` to find the information needed.

Step 1. Configure the Master Name Server

1. On the master name server, open the **/etc/named.conf** file. If there is no **/etc/named.conf** file in the **/etc** directory, create one by running the following command:

   ```
   touch /etc/named.conf
   ```

   Do the following to configure the **/etc/named.conf** file:

   a. Specify a directory clause in the options stanza. This enables the **named** data files to use paths relative to the **/usr/local/domain** directory. In this scenario, the following was added:

   ```
   options {
        directory ″/usr/local/domain″;
    };
   ```

If you choose not to specify a directory here, the **/etc** directory will be searched for the necessary data files.

b. To allow record data to be cached outside of the defined zones, specify the name of the hint zone file. In this scenario, the following was added:

```
zone ”.” IN {
     type hint;
     file ”named.ca”;
 };
```

c. Add the following stanzas to specify each zone, the type of name server you are configuring, and your name server’s domain data file. In this scenario, the master server for both forward and reverse zones is the following:

```
zone ”abc.aus.century.com” in {
     type master;
     file ”named.abc.data”;
 };
 zone ”201.9.192.in–addr.arpa” in {
     type master;
     file ”named.abc.rev”;
 };
```

d. Define the name of the **named** local file. For example:

```
zone ”0.0.127.in–addr.arpa” in {
     type master;
     file ”named.abc.local”;
 };
```

After editing the file, save and close it.

2. Open the **/usr/local/domain/named.ca** file. Add the addresses of the root name servers for the domain. The following was added in this scenario:

```
; root name servers.
.         IN    NS    relay.century.com.
 relay.century.com.   3600000    IN    A     129.114.1.2
```

After editing the file, save and close it.

3. Open the **/usr/local/domain/named.abc.local** file. Add the following information:

– The start of authority (SOA) of the zone and the default time–to–live information. The following was added in this scenario:

```
$TTL 3h    ;3 hour

 @ IN SOA venus.abc.aus.century.com. gail.zeus.abc.aus.century.com.  (

                           1       ;serial
                           3600    ;refresh
                           600     ;retry
                           3600000 ;expire
                           3600    ;negative caching TTL
 )
```

– The name server (NS) record. Insert a tab space at the beginning of the line; the **named** daemon will replace the tab space with the zone name:

```
 <tab>      IN    NS     venus.abc.aus.century.com.
```

– The pointer (PTR) record.

```
1      IN    PTR    localhost.
```

After editing the file, save and close it.

4. Open the **/usr/local/domain/named.abc.data** file. Add the following information:

– The start of authority of the zone and the default time–to–live information for the zone. This record designates the start of a zone. Only one start of authority record per zone is allowed. In this scenario, the following was added:

```
$TTL 3h    ;3 hour

 @ IN    SOA     venus.abc.aus.century.com.
gail.zeus.abc.aus.century.com. (
                 1       ;serial
                 3600    ;refresh
                 600     ;retry
                 3600000 ;expire
                 3600    ;negative caching TTL
 )
```

- The name server records for all master name servers in the zone. Insert a tab space at the beginning of the line; the **named** daemon will replace the tab space with the zone name:

```
 <tab>      IN    NS       venus.abc.aus.century.com.
```

- The name–to–address resolution information on all hosts in the name server zone of authority:

```
venus        IN    A        192.9.201.1
 earth        IN    A        192.9.201.5
 mars         IN    A        192.9.201.3
 jupiter      IN    A        192.9.201.7
```

Include other types of entries, such as canonical name records and mail exchanger records as needed. After editing the file, save and close it.

5. Open the **/usr/local/domain/named.abc.rev** file. Add the following information:

- The start of authority of the zone and the default time–to–live information. This record designates the start of a zone. Only one start of authority record per zone is allowed:

```
$TTL 3h    ;3 hour

 @  IN  SOA  venus.abc.aus.century.com. gail.zeus.abc.aus.century.com.
(
                            1       ;serial
                            3600    ;refresh
                            600     ;retry
                            3600000 ;expire
                            3600    ;negative caching TTL
 )
```

- Other types of entries, such as name server records. If you are including these records, insert a tab space at the beginning of the line; the **named** daemon will replace the tab space with the zone name. In this scenario, the following was added:

```
 <tab>      IN      NS      venus.abc.aus.century.com.
```

- Address–to–name resolution information on all hosts to be in the name server's zone of authority.

```
1                      IN    PTR    venus.abc.aus.century.com.
5                      IN    PTR    earth.abc.aus.century.com.
3                      IN    PTR    mars.abc.aus.century.com.
7                      IN    PTR    jupiter.abc.aus.century.com.
```

After editing the file, save and close it.

6. Create an **/etc/resolv.conf** file by running the following command:

```
touch /etc/resolv.conf
```

The presence of this file indicates that the host should use a name server for name resolution.

7. Add the following entry in the **/etc/resolv.conf** file:

```
nameserver 127.0.0.1
```

The `127.0.0.1` address is the loopback address, which causes the host to access itself as the name server. The **/etc/resolv.conf** file can also contain an entry similar to the following:

```
domain abc.aus.century.com
```

In this case, `abc.aus.century.com` is the domain name.After editing the file, save and close it.

8. Use the **smit stnamed** SMIT fast path to enable the **named** daemon. This initializes the daemon with each system startup. Indicate whether you want to start the **named** daemon now, at the next system restart, or both.

Step 2. Configure the Slave Name Server

To configure a slave name server, use the following procedure. You will edit a series of files and then use SMIT to start the **named** daemon.

1. On the slave name server, open the **/etc/named.conf** file. If there is no **/etc/named.conf** file in the **/etc** directory, create on by running the following command:

   ```
   touch /etc/named.conf
   ```

   Do the following to configure the **/etc/named.conf** file:

   a. Specify a directory clause in the options stanza. This enables the **named** data files to use paths relative to the **/usr/local/domain** directory. In this scenario, the following was added:

      ```
      options {
           directory ”/usr/local/domain”;
       };
      ```

      If you choose not to specify a directory here, the **named** daemon will search the **/etc** directory for the necessary data files.

   b. To allow record data to be cached outside the defined zones, specify the name of the hint zone file for the name server :

      ```
      zone ”.” IN {
           type hint;
           file ”named.ca”;
       };
      ```

   c. Specify the slave zone clauses. Each stanza includes the zone type, a file name to which the name server can back up its data, and the IP address of the master name server, from which the slave name server will replicate its data files. In this scenario, we added the following slave zone clauses:

      ```
      zone ”abc.aus.century.com” IN {
           type slave;
           file ”named.abc.data.bak”;
           masters { 192.9.201.1; };
       };

      zone ”201.9.192.in–addr.arpa” IN {
           type slave;
           file ”named.abc.rev.bak”;
           masters { 192.9.201.1; };
       };
      ```

   d. To support resolving the loopback network address, specify a zone of type *master* with a source of `named.abc.local`, as well as the domain for which the name server is responsible.

      ```
      zone ”0.0.127.in–addr.arpa” in {
           type master;
           file ”named.abc.local”;
       };
      ```

   After editing the file, save and close it.

2. Edit the **/usr/local/domain/named.ca** file.

   This file contains the address server that is the root domain server of the network. In this scenario, the following was added:

   ```
   ; root name servers.
   .          IN    NS    relay.century.com.
    relay.century.com.   3600000    IN    A     129.114.1.2
   ```

   After editing the file, save and close it.

3. Open the **/usr/local/domain/named.abc.local** file. In this scenario, the following was added:

   – The start of authority (SOA) of the zone and the default time–to–live information:

     ```
     $TTL 3h    ;3 hour

      @ IN SOA earth.abc.aus.century.com. gail.zeus.abc.aus.century.com.  (

                                  1       ;serial
                                  3600    ;refresh
                                  600     ;retry
                                  3600000 ;expire
                                  3600    ;negative caching TTL
      )
     ```

   – The name server (NS) record. Insert a tab space at the beginning of the line; the **named** daemon will replace the tab space with the zone name. For example:

     ```
      <tab>      IN    NS     earth.abc.aus.century.com.
     ```

   – The pointer (PTR) record.

     ```
     1      IN    PTR    localhost.
     ```

   After editing the file, save and close it.

4. Create an **/etc/resolv.conf** file by running the following command:

   ```
   touch /etc/resolv.conf
   ```

5. Add the following entry to that file:

   ```
   nameserver 127.0.0.1
    domain abc.aus.century.com
   ```

   After editing the file, save and close it.

6. Use the **smit stnamed** SMIT fast path to enable the **named** daemon. This initializes the daemon with each system startup. Indicate whether you want to start the **named** daemon now, at the next system restart, or both.

Step 3. Configure the Hint Name Server

To configure a hint, or *cache–only*, name server, use the following procedure, which edits a series of files and then uses SMIT or the command line to start the **named** daemon.

1. On the hint name server, edit the **/etc/named.conf** file. If there is no **/etc/named.conf** file in the **/etc** directory, create one by running the following command:

   ```
   touch /etc/named.conf
   ```

   Do the following to configure the **/etc/named.conf** file:

   a. Specify a directory clause in the options stanza. This enables the **named** data files to use paths relative to the **/usr/local/domain** directory. In this scenario, the following was added:

      ```
      options {
           directory ”/usr/local/domain”;
       };
      ```

b. To support resolving the loopback network address, specify a zone of type *master* with a source of **named.abc.local**, as well as the domain for which the name server is responsible. In this example, the options directory keyword was specified in the **/etc/named.conf** file.

```
zone ”0.0.127.in–addr.arpa” IN {
     type master;
       file ”named.abc.local”;
 };
```

c. Specify the name of the cache zone file. For example:

```
zone ”.” IN {
     type hint;
     file ”named.ca”;
 };
```

After editing the file, save and close it.

2. Edit the **/usr/local/domain/named.ca** file.

   This file contains the addresses of the servers that are authoritative name servers for the root domain of the network. For example:

```
; root name servers.
.          IN    NS    relay.century.com.
 relay.century.com.   3600000    IN    A     129.114.1.2
```

   After editing the file, save and close it.

3. Edit the **/usr/local/domain/named.local** file. In this scenario, the following information was added to this file:

   – The start of authority (SOA) of the zone and the default time–to–live information:

```
$TTL 3h    ;3 hour

 @ IN SOA venus.abc.aus.century.com. gail.zeus.abc.aus.century.com.  (

                            1       ;serial
                            3600    ;refresh
                            600     ;retry
                            3600000 ;expire
                            3600    ;negative caching TTL
 )
```

   – The name server (NS) record. Insert a tab space at the beginning of the line; the **named** daemon will replace the tab space with the zone name:

```
 <tab>      IN    NS     venus.abc.aus.century.com.
```

   – The pointer (PTR) record.

```
1      IN    PTR    localhost.
```

   After editing the file, save and close it.

4. Create an **/etc/resolv.conf** file by running the following command:

```
touch /etc/resolv.conf
```

5. Add the following entry to that file:

```
nameserver 127.0.0.1
 domain abc.aus.century.com
```

   After editing the file, save and close it.

6. Use the **smit stnamed** SMIT fast path to enable the **named** daemon. This initializes the daemon with each system startup. Indicate whether you want to start the **named** daemon now, at the next system restart, or both.

Chapter 2. Communications and Networks Overview

This chapter presents the conceptual foundation for understanding computer networking in general. System administrators unfamiliar with general networking principles need to read this chapter. Those familiar with UNIX networking can safely skip this chapter.

A network is the combination of two or more computers and their connecting links. A *physical* network is the hardware (equipment such as adapter cards, cables, and telephone lines) that makes up the network. The software and the conceptual model make up the *logical* network.

This overview provides the following information on networks:

- Communications Functions Introduction on page 2-2
- Network Introduction on page 2-3
- Physical Networks Introduction on page 2-5
- Network Systems and Protocols on page 2-6
- Communicating with Other Operating Systems on page 2-8

Communications Functions Introduction

Networks allow for several user and application communication functions. For example, they enable a user to do the following:

- Send electronic mail (e–mail)
- Emulate another terminal or log in to another computer
- Transfer data
- Run programs that reside on a remote node.

One of the most popular applications for computer networks is e–mail, which allows a user to send a message to another user. The two users may be on the same system (in which case a communications network is not needed), different systems in different buildings, or even in different countries.

Through a communications network, one computer can mimic another and access information as if it were a different type of computer or terminal. Remote login capabilities allow users to log in to a remote system and access the same programs and files as if they were using the machine locally.

Networks also allow for the transfer of data from one system to another. Files, directories, and entire file systems can be migrated from one machine to another across a network, enabling remote backup of data, as well as assuring redundancy in case of machine failure.

Several different protocols exist that allow users and applications on one system to invoke procedures and applications on other systems, which is useful when distributing the burden for computer–intensive routines.

Network Introduction

The complexity of modern computer networks has given rise to several conceptual models for explaining how networks work. One of the most common of these models is the International Standards Organization's Open Systems Interconnection (OSI) Reference Model, also referred to as the OSI seven–layer model. The seven layers of the OSI model are numbered as follows:

7	**Application**
6	**Presentation**
5	**Session**
4	**Transport**
3	**Network**
2	**Data Link**
1	**Physical**

Levels 1 through 3 are network–specific, and differ depending on what physical network you are using. Levels 4 through 7 comprise network–independent, higher–level functions. Each layer describes a particular function (instead of a specific protocol) that occurs in data communications. The seven layers function from lowest level (machine level) to highest level (the level at which most human interaction takes place), as follows:

Application
Comprises the applications that use the network.

Presentation
Ensures that data is presented to the applications in a consistent fashion.

Session
Manages the connections between applications.

Transport
Ensures error–free data transmission.

Network
Manages the connections to other machines on the network.

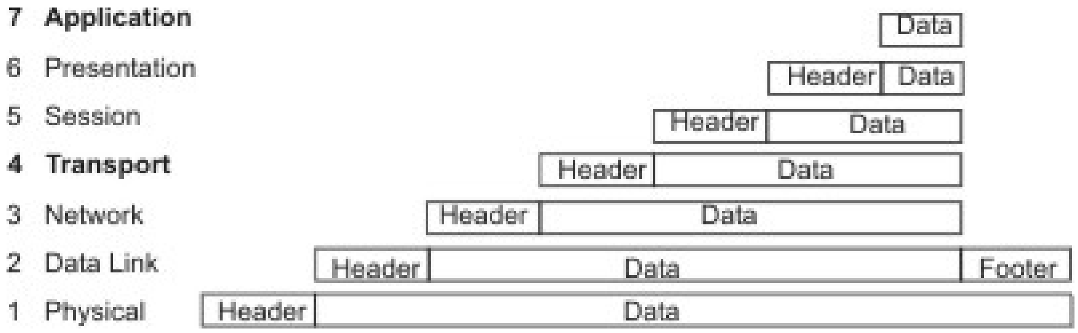
Data Link
Provides reliable delivery of data across the physical layer (which is usually inherently unreliable).

Physical
Describes the physical media of the network. For example, the fiber optic cable required for a Fiber Distributed Data Interface (FDDI) network is part of the physical layer.

Note:: While the OSI Reference Model is useful for discussing networking concepts, many networking protocols do not closely follow the OSI model. For example, when discussing Transmission Control Protocol/Internet Protocol (TCP/IP), the Application and Presentation layer functions are combined, as are the Session and Transport layers and the Data Link and Physical layers.

Each layer in the OSI model communicates with the corresponding layer on the remote machine as shown in the OSI Reference Model figure.

Figure 1. OSI Reference Model This illustration shows the various communication levels of the OSI Model as described in the above text.



The layers pass data only to the layers immediately above and below. Each layer adds its own header information (and footer information, in the case of the Data Link), effectively encapsulating the information received from the higher layers.

Individual users as well as organizations use networks for many reasons, including:

- Data entry
- Data queries
- Remote batch entry
- Resource sharing
- Data sharing
- Electronic mail.

Data entry consists of entering data directly into either local or remote data files. Increased accuracy and efficiency are natural by–products of a one–step data transfer. Data queries entail searching data files for specified information. Data updating involves altering, adding, or deleting data stored in local or remote files. Remote batch entry consists of entering batches of data from a remote location, an activity often performed at night or during periods of low system usage. Because of such diverse capabilities, communications and networks are not only desirable but necessary.

Sharing resources is another function of networks. Users can share data as well as programs, file–storage space, and peripheral devices like printers, modems, terminals, and fixed disks. Sharing of system resources is cost effective because it eliminates the problems of keeping multiple copies of programs and it keeps data consistent (in the case of program and file sharing).

Physical Networks Introduction

The physical network consists of the cables (coaxial cable, twisted pair, fiber optic, and telephone lines) that connect the different hardware residing on the network, the adapter cards used on the attached hosts, and any concentrators, repeaters, routers, or bridges used in the network. (A *host* is a computer attached to the network.)

Physical networks vary both in size and in the type of hardware used. The two common kinds of networks are *local area networks* (LANs) and *wide area networks* (WANs). A LAN is a network where communications are limited to a moderately sized geographic area of 1 to 10 km (1 to 6 miles), such as a single office building, warehouse, or campus. A WAN is a network providing data communications capability throughout geographic areas larger than those serviced by LANs, such as across a country or across continents. An intermediate class of networks exists also, called *metropolitan area networks* (MANs). This guide does not generally distinguish MANs; they are grouped with WANs.

LANs commonly use Standard Ethernet, IEEE 802.3 Ethernet, or token–ring hardware for the physical network, while WANs and asynchronous networks use communications networks provided by common carrier companies. Operation of the physical network in both cases is usually controlled by networking standards from organizations such as the Electronics Industry Association (EIA) or the International Telecommunication Union (ITU).

Network Systems and Protocols

All network communications involve the use of hardware and software. *Hardware* consists of the physical equipment connected to the physical network. *Software* consists of the programs and device drivers pertaining to the operation of a particular system.

The system hardware consists of adapter cards or other devices that provide a path or interface between the system software and the physical network. An adapter card requires an input/output (I/O) card slot in the system. Other devices, such as modems, can be attached to one of the standard ports on the computer.

Adapter cards support the standards required by the physical network (for example, EIA 232D, Smartmodem, V.25 bis, EIA 422A, X.21, or V.35) and may, at the same time, support software *protocols*, for example, synchronous data link control (SDLC), high–level data link control (HDLC), and bisynchronous protocols. If the adapter does not contain software support, then this support must be provided by the adapter device driver.

Protocols

All communications software use *protocols*, sets of semantical and syntactical rules that determine the behavior of functional units in achieving communication. Protocols define how information is delivered, how it is enclosed to reach its destination safely, and what path it follows. Protocols also coordinate the flow of messages and their acknowledgments.

Protocols exist at different levels within the kernel and cannot be manipulated directly. However, they are manipulated indirectly by what the user chooses to do at the application programming interface (API) level. The choices a user makes when invoking file transfer, remote login, or terminal emulation programs define the protocols used in the execution of those programs.

Addresses

Addresses are associated with both software and hardware. The address is the means by which the sending or control station selects the station to which it sends data. Addresses identify receiving or storage locations. A physical address is a unique code assigned to each device or workstation connected to a network.

For example, on a token–ring network, the **netstat –iv** command displays the token–ring card address. This is the physical network address. The **netstat –iv** command also displays class–level and user–level address information. Addresses are often defined by software but can be created by the user as well.

Domains

An aspect of addresses common to many communications networks is the concept of *domains*. For example, the structure of the Internet illustrates how domains define the Internet Protocol (IP) address. The Internet is an extensive network made up of many different smaller networks. To facilitate routing and addressing, Internet addresses are hierarchically structured in domains, with very broad categories at the top such as `com` for commercial users, `edu` for educational users, and `gov` for government users.

Within the `com` domain are many smaller domains corresponding to individual businesses; for example, `ibm`. Within the `ibm.com` domain are even smaller domains corresponding to the Internet addresses for various locations, such as `austin.ibm.com` or `raleigh.ibm.com`. At this level, we start seeing names of *hosts*. A host, in this context, is any computer connected to the network. Within `austin.ibm.com`, there may be hosts with the names `hamlet` and `lear`, which are addressed `hamlet.austin.ibm.com` and `lear.austin.ibm.com`.

Gateways and Bridges

A wide variety of networks reside on the Internet, often using different hardware and running different software. *Gateways* and *bridges* enable these different networks to communicate with each other. A bridge is a functional unit that connects two LANs that possibly use the same logical link control (LLC) procedure, such as Ethernet, but different medium access control (MAC) procedures. A gateway has a broader range than a bridge. It operates above the link layer and, when required, translates the interface and protocol used by one network into those used by another distinct network. Gateways allow data transfers across the various networks that constitute the Internet.

Routing

Using domain names for addressing and gateways for translation greatly facilitates the *routing* of the data being transferred. Routing is the assignment of a path by which a message reaches its destination. The domain name effectively defines the message destination. In a large network like the Internet, information is routed from one communications network to the next until that information reaches its destination. Each communications network checks the domain name and, based on the domains with which that network is familiar, routes the information on to the next logical stop. In this way, each communications network that receives the data contributes to the routing process.

Local and Remote Nodes

A physical network is used by the hosts that reside on that network. Each host is a *node* on the network. A node is an addressable location in a communications network that provides host–processing services. The intercommunication of these various nodes are defined as *local* or *remote*. *Local* pertains to a device, file, or system accessed directly from your system, without the use of a communications line. *Remote* pertains to a device, file, or system accessed by your system over a communications line. Local files reside on your system, while remote files reside on a file server or at another node with which you communicate using a physical network, for example, Ethernet, token–ring, or phone lines.

Client and Server

Related to the concepts of local and remote are those of *client* and *server.* A server is a computer that contains data or provides facilities to be accessed by other computers on the network. Common server types are file servers, which store files; name servers, which store names and addresses; and application servers, which store programs and applications; print servers, which schedule and direct print jobs to their destination.

A client is a computer requesting services or data from a server. A client, for example, could request updated program code or the use of applications from a code server. To obtain a name or address, a client contacts a name server. A client could also request files and data for data entry, inquiry, or record updating from a file server.

Communicating with Other Operating Systems

Different types of computers can be connected on a network. The computers can be from different manufacturers or be different models from the same manufacturer. Communication programs bridge the differences in operating systems of two or more types of computers.

Sometimes these programs require that another program has previously been installed on the network. Other programs may require that such communications connectivity protocols as TCP/IP or Systems Network Architecture (SNA) exist on the network.

For example, with AIX 4.3.2 and later, AIX Fast Connect for Windows lets PC clients access operating system files and printers using native PC networking client software. PC users can use remote operating system file systems directly from their machines as if they were locally stored. They can print jobs on printers using the operating system spooler, view available printers, and map a printer as a network printer. For more information on AIX Fast Connect, see *AIX Fast Connect for Windows Version 3.1 Guide*.

Chapter 3. Mail

The mail facility provides a method for exchanging electronic mail (e–mail) with users on the same system or on multiple systems connected by a network. This section documents the mail system, the standard mail user interface, the Internet Message Access Protocol (IMAP), and the Post Office Protocol (POP).

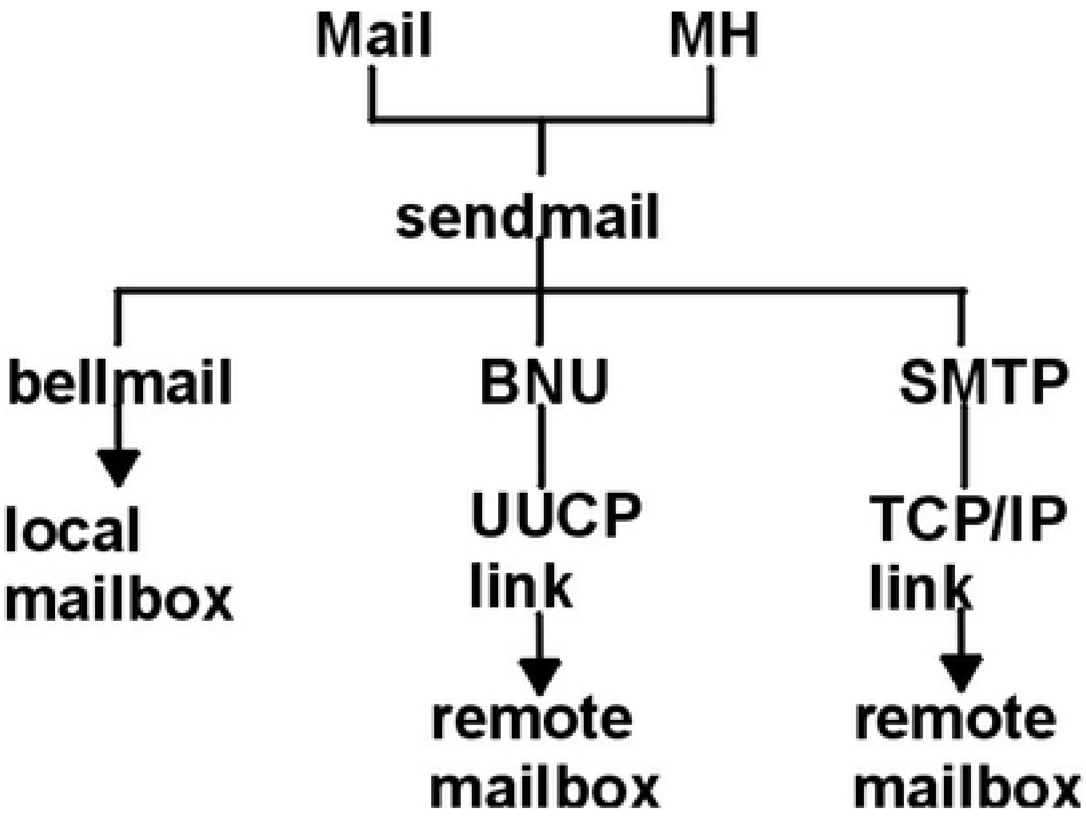
The mail system is an internetwork mail delivery facility that consists of a user interface, a message routing program, and a message delivery program (or mailer). The mail system relays messages from one user to another on the same host, between hosts, and across network boundaries. It also performs a limited amount of message–header editing to put the message into a format that is appropriate for the receiving host.

A mail *user interface* enables users to create and send messages to, and receive messages from, other users. The mail system provides two user interfaces, **mail** and **mhmail**. The **mail** command is the standard mail user interface available on all UNIX systems. The **mhmail** command is the Message Handler (MH) user interface, an enhanced mail user interface designed for experienced users.

A *message routing program* routes messages to their destinations. The mail system message routing program is the **sendmail** program, which is part of the Base Operating System (BOS) and is installed with BOS. The **sendmail** program is a daemon that uses information in the **/etc/mail/sendmail.cf** file, the **/etc/mail/aliases** file to perform the necessary routing.

Note:: In versions earlier than AIX 5.1, the **sendmail.cf** and **aliases** files are located in **/etc/sendmail.cf** and **/etc/aliases**, respectively.

Depending on the type of route to the destination, the **sendmail** command uses different *mailers* to deliver messages.



As the figure illustrates:

- To deliver local mail, the **sendmail** program routes messages to the **bellmail** program. The **bellmail** program delivers all local mail by appending messages to the user’s system mailbox, which is in the **/var/spool/mail** directory.
- To deliver mail over a UNIX–to–UNIX Copy Program (UUCP) link, the **sendmail** program routes messages using Basic Network Utilities (BNU).
- To deliver mail routed through Transmission Control Protocol/Internet Protocol (TCP/IP), the **sendmail** command establishes a TCP/IP connection to the remote system then uses Simple Mail Transfer Protocol (SMTP) to transfer the message to the remote system.

Mail Management Tasks

The following is a list of the tasks for which you, the mail manager, are responsible.

1. Configure the **/etc/rc.tcpip** file so that the **sendmail** daemon will be started at system boot time. See Configuring the /etc/rc.tcpip File to Start the sendmail Daemon.
2. Customize the configuration file **/etc/mail/sendmail.cf**. The default **/etc/mail/sendmail.cf** file is configured so that both local mail and TCP/IP mail can be delivered. In order to deliver mail through BNU, you must customize the **/etc/mail/sendmail.cf** file. See the sendmail.cf File in *AIX 5L Version 5.2 Files Reference* for more information.
3. Define system–wide and domain–wide mail aliases in the **/etc/mail/aliases** file. See Managing Mail Aliases for more information.
4. Manage the Mail Queue. See Managing the Mail Queue Files and Directories for more information.
5. Manage the Mail Log. See Managing Mail Logging for more information.

Configuring the /etc/rc.tcpip File to Start the sendmail Daemon

To configure the **/etc/rc.tcpip** file so that the **sendmail** daemon will be started at system boot time:

1. Edit the **/etc/rc.tcpip** file with your favorite text editor.
2. Find the line that begins with **start /usr/lib/sendmail**. By default, this line should be uncommented, that is, there is no # (pound sign) at the beginning of the line. However, if it is commented, delete the pound sign.
3. Save the file.

With this change, the system will start the **sendmail** daemon at boot time.

Managing Mail Aliases

Aliases map names to address lists using personal, system–wide, and domain–wide alias files. You can define three types of aliases:

personal

Defined by individual users in the user's **$HOME/.mailrc** file.

local system

Defined by the mail system administrator in the **/etc/mail/aliases** file. These aliases apply to mail handled by the **sendmail** program on the local system. Local system aliases rarely need to be changed.

domainwide

By default, **sendmail** reads **/etc/alias** to resolve aliases. To override the default and use NIS, edit or create **/etc/netsvc.conf** and add the line:

```
aliases=nis
```

/etc/mail/aliases File

Note:: In versions earlier than AIX 5.1, the **aliases** file is located in **/etc/aliases**.

The **/etc/mail/aliases** file consists of a series of entries in the following format:

```
 Alias :   Name1 ,   Name2 , ...   NameX
```

where *Alias* can be any alphanumeric string that you choose (not including special characters, such as @ or !). *Name1* through *NameX* is a series of one or more recipient names. The list of names can span one or more lines. Each continued line begins with a space or a tab. Blank lines and lines beginning with a # (pound sign) are comment lines.

The **/etc/mail/aliases** file must contain the following three aliases:

MAILER–DAEMON

The ID of the user who is to receive messages addressed to the mailer daemon. This name is initially assigned to the root user:

```
MAILER–DAEMON: root
```

postmaster

The ID of the user responsible for the operation of the local mail system. The **postmaster** alias defines a single mailbox address that is valid at each system in a network. This address enables users to send inquiries to the **postmaster** alias at any system, without knowing the correct address of any user at that system. This name is initially assigned to the root user:

```
postmaster: root
```

nobody

The ID that is to receive messages directed to programs such as **news** and **msgs**. This name is initially assigned to **/dev/null:**

```
nobody: /dev/null
```

To receive these messages, define this alias to be a valid user.

Whenever you change this file, you must recompile it into a database format that the **sendmail** command can use. See Building the Alias Database.

Creating Local System Aliases for Mail

For step–by–step instructions on how to create a local system alias, see Create a Local Alias for Mail on page 1-19.

Building the Alias Database

The **sendmail** command does not use directly the alias definitions in the local system **/etc/mail/aliases** file. Instead, the **sendmail** command reads a processed database manager (dbm) version of the **/etc/mail/aliases** file. You can compile the alias database using one of the following methods:

- Run the **/usr/sbin/sendmail** command using the **–bi** flag.
- Run the **newaliases** command. This command causes the **sendmail** command to read the local system **/etc/mail/aliases** file and create a new file containing the alias database information. This file is in the more efficient Berkeley format:

  /etc/mail/aliases.db

  (Versions earlier than AIX 5.1 created two database files, **/etc/aliases.dir** and **/etc/aliases.pag**.)
- Run the **sendmail** command using the **Rebuild Aliases** flag. This rebuilds the alias database automatically when it is out–of–date. Auto–rebuild can be dangerous on heavily loaded machines with large alias files. If it might take more than the rebuild time–out (normally five minutes) to rebuild the database, there is a chance that several processes will start the rebuild process simultaneously. Notes:

1. If these files do not exist, the **sendmail** command cannot process mail and will generate an error message.
2. If you have multiple alias databases specified, the **–bi** flag rebuilds all the database types it understands (for example, it can rebuild Network Database Management (NDBM) databases but not NIS databases).

The **/etc/netsvc.conf** file contains the ordering of system services. To specify the service ordering of aliases, add the following line:

```
aliases=service, service
```

where `service` can be either `files` or `nis`. For example:

```
aliases=files, nis
```

tells the **sendmail** command to try the local alias file first; and if that fails, try `nis`. If `nis` is defined as a service, it should be running.

For further information on the **/etc/ netsvc.conf** file, see *AIX 5L Version 5.2 Files Reference*.

Managing the Mail Queue Files and Directories

The mail queue is a directory that stores data and controls files for mail messages that the **sendmail** command delivers. By default, the mail queue is **/var/spool/mqueue**.

Mail messages might be queued for many reasons.

For example:

1. The **sendmail** command can be configured to process the queue at certain intervals, rather than immediately. If this is so, mail messages must be stored temporarily.
2. If a remote host does not answer a request for a mail connection, the mail system queues the message and tries again later.

Printing the Mail Queue

The contents of the queue can be printed using the **mailq** command (or by specifying the **–bp** flag with the **sendmail** command).

These commands produce a listing of the queue IDs, the sizes of the messages, the dates the messages entered the queue, and the senders and recipients.

Mail Queue Files

Each message in the queue has a number of files associated with it. The files are named according to the following conventions:

```
 Type  f   ID
```

where *ID* is a unique message queue ID, and *Type* is one of the following letters indicating the type of file:

d The data file containing the message body without the heading information.

q The queue–control file. This file contains the information necessary to process the job.

t A temporary file. This file is an image of the **q** file when it is being rebuilt. It is quickly renamed to the **q** file.

x A transcript file that exists during the life of a session and shows everything that happens during that session.

For example, if a message has a queue ID of AA00269, the following files are created and deleted in the mail queue directory while the **sendmail** command tries to deliver the message:

dfAA00269	Data file
qfAA00269	Control file
tfAA00269	Temporary file
xfAA00269	Transcript file

q Control File

The **q** control file contains a series of lines, each beginning with a code letter:

B Specifies the `body type`. The remainder of the line is a text string defining the `body type`. If this entire field is missing, the `body type` is 7–bit by default, and no special processing is attempted. Legal values are **7BIT** and **8BITMIME**.

C Contains the controlling user. For recipient addresses that are a file or a program, **sendmail** performs delivery as the owner of the file or program. The controlling user is set to the owner of the file or program. Recipient addresses that are read from a**.forward** or**:include:** file will also have the controlling user set to the owner of the file. When **sendmail** delivers mail to these recipients, it delivers as the controlling user, then converts back to root.

F Contains envelope flags. The flags are any combination of **w**, which sets the **EF_WARNING** flag; **r**, which sets the **EF_RESPONSE** flag; **8**, which sets the **EF_HAS8BIT** flag; and **b**, which sets the **EF_DELETE_BCC** flag. Other letters are silently ignored.

H Contains a heading definition. There can be any number of these lines. The order in which the **H** lines appear determines their order in the final message. These lines use the same syntax as heading definitions in the **/etc/mail/sendmail.cf** configuration file. (For versions earlier than AIX 5.1, this file is **/etc/sendmail.cf**.)

I Specifies the inode and device information for the **df** file; this can be used to recover your mail queue after a disk crash.

K Specifies the time (as seconds) of the last delivery attempt.

M When a message is put into the queue because an error occurred during a delivery attempt, the nature of the error is stored in the **M** line.

N Specifies the total number of delivery attempts.

O Specifies the original message transfer system (MTS) value from the ESMTP. It is used for Delivery Status Notifications only.

P Contains the priority of the current message. The priority is used to order the queue. Higher numbers mean lower priorities. The priority increases as the message sits in the queue. The initial priority depends on the message class and the size of the message.

Q Contains the original recipient as specified by the `ORCPT=` field in an ESMTP transaction. Used exclusively for Delivery Status Notifications. It applies only to the immediately following **R** line.

R Contains a recipient address. There is one line for each recipient.

S Contains the sender address. There is only one of these lines.

T Contains the message creation time used to compute when to time out the message.

V Specifies the version number of the queue file format used to allow new **sendmail** binaries to read queue files created by older versions. Defaults to version **zero**. Must be the first line of the file, if present.

Z Specifies the original envelope ID (from the ESMTP transaction). Used for Delivery Status Notifications only.

$** Contains a macro definition. The values of certain macros (**$r and **$s**) are passed through to the queue run phase.

The **q** file for a message sent to `amy@zeus` would look similar to:

```
P217031
 T566755281
 MDeferred: Connection timed out during user open with zeus
 Sgeo
 Ramy@zeus
 H?P?return–path: <geo>
 Hreceived: by george (0.13 (NL support)/0.01)
         id AA00269; Thu, 17 Dec 87 10:01:21 CST
 H?D?date: Thu, 17 Dec 87 10:01:21 CST
 H?F?From: geo
 Hmessage–id: <8712171601.AA00269@george>
 HTo: amy@zeus
 Hsubject: test
```

Where:

`P217031`	Priority of the message
`T566755281`	Submission time in seconds
`MDeferred: Connection timed out during user open with zeus`	Status message
`Sgeo`	ID of the sender
`Ramy@zeus`	ID of the receiver
`H lines`	Header information for the message

Specifying Time Values in sendmail

To set the message time–out and queue processing interval, you must use a specific format for the time value. The format of a time value is:

–q   *NumberUnit*

where *Number* is an integer value and *Unit* is the unit letter. *Unit* can have one of the following values:

s	Seconds
m	Minutes
h	Hours
d	Days
w	Weeks

If *Unit* is not specified, the **sendmail** daemon uses minutes (**m**) as the default. Here are three examples illustrating time–value specification:

```
/usr/sbin/sendmail –q15d
```

This command tells the **sendmail** daemon to process the queue every 15 days.

```
/usr/sbin/sendmail –q15h
```

This command tells the **sendmail** daemon to process the queue every 15 hours.

```
/usr/sbin/sendmail –q15
```

This command tells the **sendmail** daemon to process the queue every 15 minutes.

Forcing the Mail Queue

In some cases, you might find that the queue is clogged for some reason. You can force a queue to run using the **–q** flag (with no value). You can also use the **–v** flag (verbose) to watch what happens:

```
/usr/sbin/sendmail –q –v
```

You can also limit the jobs to those with a particular queue identifier, sender, or recipient using one of the queue modifiers. For example, **–qRsally** restricts the queue run to jobs that have the string **sally** in one of the recipient addresses. Similarly, **–qS** *string* limits the run to particular senders, and **–qI** *string* limits it to particular queue identifiers.

Setting the Queue Processing Interval

The value of the **–q** flag when the daemon starts determines the interval at which the **sendmail** daemon processes the mail queue.

The **sendmail** daemon is usually started by the **/etc/rc.tcpip** file, at system startup. The **/etc/rc.tcpip** file contains a variable called the queue processing interval (QPI), which it uses to specify the value of the **–q** flag when it starts the **sendmail** daemon. By default, the value of **qpi** is 30 minutes. To specify a different queue processing interval:

1. Edit the **/etc/rc.tcpip** file with your favorite editor.
2. Find the line that assigns a value to the *qpi* variable, such as:

   ```
   qpi=30m
   ```
3. Change the value assigned to the *qpi* variable to the time value you prefer.

These changes will take effect at the next system restart. If you want the changes to take effect immediately, stop and restart the **sendmail** daemon, specifying the new **–q** flag value. See Stopping the sendmail Daemon on page 3-10 and Starting the sendmail Daemon on page 3-10 for more information.

Moving the Mail Queue

When a host goes down for an extended period, many messages routed to (or through) that host might be stored in your mail queue. As a result, the **sendmail** command spends a long time sorting the queue, severely degrading your system performance. If you move the queue to a temporary place and create a new queue, the old queue can be run later when the host returns to service. To move the queue to a temporary place and create a new queue:

1. Stop the **sendmail** daemon by following the instructions in Stopping the sendmail Daemon.
2. Move the entire queue directory by entering:

   ```
   cd /var/spool
    mv mqueue omqueue
   ```
3. Restart the **sendmail** daemon by following the instructions in Starting the sendmail Daemon.
4. Process the old mail queue by entering:

   ```
   /usr/sbin/sendmail –oQ/var/spool/omqueue –q
   ```

   The **–oQ** flag specifies an alternate queue directory. The **–q** flag specifies to run every job in the queue. To get a report about the progress of the operation, use the **–v** flag.

   Note:: This operation can take some time.
5. Remove the log files and the temporary directory when the queue is empty by entering:

   ```
   rm /var/spool/omqueue/*
    rmdir /var/spool/omqueue
   ```

Starting the sendmail Daemon

To start the **sendmail** daemon, enter either of the following commands:

```
startsrc –s sendmail –a ”–bd –q15”
```

```
/usr/lib/sendmail –bd –q15
```

If the **sendmail** daemon is already active when you enter one of these commands, you see the following message on the screen:

```
The sendmail subsystem is already active. Multiple instances are not
supported.
```

If the **sendmail** daemon is not already active, then you see a message indicating that the **sendmail** daemon has been started.

Stopping the sendmail Daemon

To stop the **sendmail** daemon, run the **stopsrc –s sendmail** command.

If the **sendmail** daemon was not started with the **startsrc** command:

- Find the **sendmail** process ID.
- Enter the **kill** *sendmail_pid* command (where *sendmail_pid* is the process ID of the **sendmail** process).

Managing Mail Logging

The **sendmail** command logs mail system activity through the **syslogd** daemon. The **syslogd** daemon must be configured and running for logging to occur. Specifically, the **/etc/syslog.conf** file should contain the uncommented line:

```
mail.debug          /var/spool/mqueue/log
```

If it does not, use your favorite editor to make this change; be certain that the path name is correct. If you change the **/etc/syslog.conf** file while the **syslogd** daemon is running, refresh the **syslogd** daemon by typing the following command at a command line:

```
refresh –s syslogd
```

If the **/var/spool/mqueue/log** file does not exist, you must create it by typing the following command:

```
touch /var/spool/mqueue/log
```

Messages in the log file appear in the following format:

Each line in the system log consists of a time stamp, the name of the machine that generated it (for logging from several machines over the local area network), the word `sendmail:`, and a message. Most messages are a sequence of *name=value* pairs.

The two most common lines logged when a message is processed are the **receipt** line and the **delivery attempt** line. The **receipt** line logs the receipt of a message; there will be one of these per message. Some fields may be omitted. These message fields are:

`from`	Specifies the envelope sender address.
`size`	Specifies the size of the message in bytes.
`class`	Indicates the class (numeric precedence) of the message.
`pri`	Specifies the initial message priority (used for queue sorting).
`nrcpts`	Indicates the number of envelope recipients for this message (after aliasing and forwarding).
`proto`	Specifies the protocol used to receive the message, for example ESMTP or UNIX–to–UNIX Copy Program (UUCP).
`relay`	Specifies the machine from which it was received.

The **delivery attempt** line is logged each time there is delivery attempt (so there can be several per message if delivery is deferred or there are multiple recipients). These fields are:

`to`	Contains a comma–separated list of the recipients to this mailer.
`ctladdr`	Specifies the *controlling user*, that is, the name of the user whose credentials are used for delivery.
`delay`	Specifies the total delay between the time this message was received and the time it was delivered.
`xdelay`	Specifies the amount of time needed in this delivery attempt.
`mailer`	Specifies the name of the mailer used to deliver to this recipient.
`relay`	Specifies the name of the host that actually accepted (or rejected) this recipient.
`stat`	Specifies the delivery status.

Because such a large amount of information can be logged, the log file is arranged as a succession of levels. Beginning at level 1, the lowest level, only very unusual situations are logged. At the highest level, even the insignificant events are logged. As a convention, log levels ten and under the most useful information. Log levels above 64 are reserved for debugging purposes. Levels from 11–64 are reserved for verbose information.

The types of activities that the **sendmail** command puts into the log file are specified by the **L** option in the **/etc/mail/sendmail.cf** file. (For versions earlier than AIX 5.1, this file is **/etc/sendmail.cf**.)

Managing the Log

Because information is continually appended to the end of the log, the file can become very large. Also, error conditions can cause unexpected entries to the mail queue. To keep the mail queue and the log file from growing too large, run the **/usr/lib/smdemon.cleanu** shell script. This script forces the **sendmail** command to process the queue and maintains four progressively older copies of log files, named **log.0**, **log.1**, **log.2**, and **log.3**. Each time the script runs it moves:

- **log.2** to **log.3**
- **log.1** to **log.2**
- **log.0** to **log.1**
- **log** to **log.0**

Running this script allows logging to start over with a new file. Run this script either manually or at a specified interval with the **cron** daemon.

Logging Traffic

Many Simple Mail Transfer Protocols (SMTPs) implementations do not fully implement the protocol. For example, some personal computer–based SMTPs do not understand continuation lines in reply codes. These can be very hard to trace. If you suspect such a problem, you can set traffic logging by using the **–X** flag. For example:

```
/usr/sbin/sendmail –X /tmp/traffic –bd
```

This command logs all traffic in the **/tmp/traffic** file.

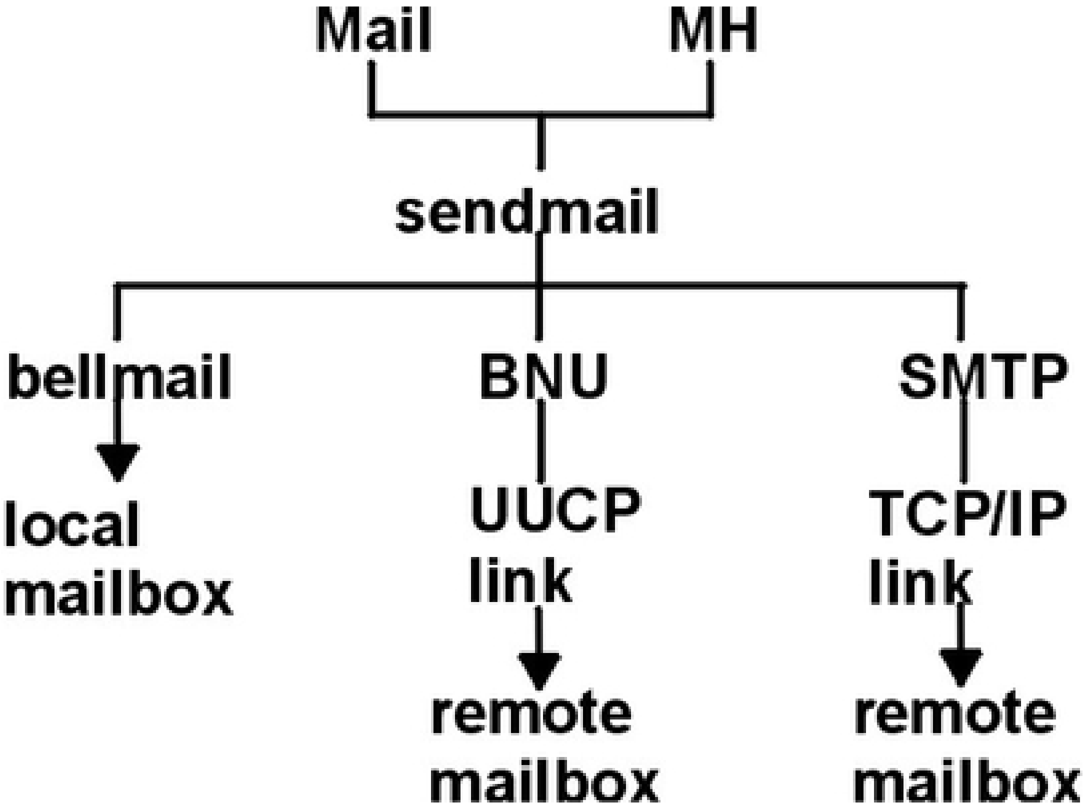
Because this command logs a lot of data very quickly, it should never be used during normal operations. After running the command, force the **errant** implementation to send a message to your host. All message traffic in and out of **sendmail**, including the incoming SMTP traffic, will be logged in this file.

Using **sendmail**, you can log a dump of the open files and the connection cache by sending it a **SIGUSR1** signal. The results are logged at **LOG_DEBUG** priority.

Logging Mailer Statistics

The **sendmail** command tracks the volume of mail being handled by each of the mailer programs that interface with it. Those mailers are defined in the **/etc/mail/sendmail.cf** file. (For versions earlier than AIX 5.1, this file is **/etc/sendmail.cf**.)

Figure 2. Mailers Used by the Sendmail Command This illustration is a type of top–down organizational chart with Mail and MH at the top. Branching from them are bellmail, BNU and SMTP. Underneath the previous level are local mailbox, UUCP link, and TCP/IP link respectively. Beneath UUCP link is remote mailbox and under TCP/IP link is remote mailbox.



To start the accumulation of mailer statistics, create the **/etc/mail/statistics** file by typing the following:

```
touch /etc/mail/statistics
```

If the **sendmail** command encounters errors when trying to record statistics information, the command writes a message through the **syslog** subroutine. These errors do not affect other operations of the **sendmail** command.

The **sendmail** command updates the information in the file each time it processes mail. The size of the file does not grow, but the numbers in the file do. They represent the mail volume since the time you created or reset the **/etc/mail/statistics** file.

Note:: In versions earlier than AIX 5.1, statistics were kept in the **/var/tmp/sendmail.st** file.

Displaying Mailer Information

The statistics kept in the **/etc/mail/statistics** file are in a database format that cannot be read as a text file. To display the mailer statistics, type the following at a command prompt:

```
/usr/sbin/mailstats
```

This reads the information in the **/etc/mail/statistics** file, formats it, and writes it to standard output. For information on the output of the **/usr/sbin/mailstats** command, read its description in the *AIX 5L Version 5.2 Commands Reference*.

Debugging sendmail

There are a large number of debug flags built into the **sendmail** command. Each debug flag has a number and level, where higher levels print more information. The convention is levels greater than nine print out so much information that they are used only for debugging a particular piece of code. Debug flags are set using the **–d** flag as shown in the following example:

```
debug–flag:     –d debug–list
debug–list:     debug–flag[.debug–flag]*
debug–flag:     debug–range[.debug–level]
debug–range:    integer|integer–integer
debug–level:    integer

–d12            Set flag 12 to level 1
–d12.3          Set flag 12 to level 3
–d3–17          Set flags 3 through 17 to level 1
–d3–17.4        Set flags 3 through 17 to level 4
```

The available debug flags are:

–d0	General debugging.
–d1	Show send information.
–d2	End with *finis* ().
–d3	Print the load average.
–d4	Enough disk space.
–d5	Show events.
–d6	Show failed mail.
–d7	The queue file name.
–d8	DNS name resolution.
–d9	Trace RFC1413 queries.
–d9.1	Make host name canonical.
–d10	Show recipient delivery.
–d11	Trace delivery.
–d12	Show mapping of relative host.
–d13	Show delivery.
–d14	Show header field commas.
–d15	Show network get request activity.
–d16	Outgoing connections.
–d17	List MX hosts.

Note: There are now almost 200 defined debug flags in **sendmail**.

Internet Message Access Protocol (IMAP) and Post Office Protocol (POP)

AIX provides two Internet–based mail protocol server implementations for accessing mail remotely:

- Post Office Protocol (POP)
- Internet Message Access Protocol (IMAP)

Both the POP and IMAP servers store and provide access to electronic messages. Using these mail access protocols on a server eliminates the requirement that, to receive mail, a computer must always be up and running.

The POP server provides an off–line mail system, whereby a client, using POP client software, can remotely access a mail server to retrieve mail messages. The client can either download the mail messages and immediately delete the messages from the server, or download the messages and leave the messages resident on the POP server. After the mail is downloaded to the client machine, all mail processing is local to the client machine. The POP server allows access to a user mailbox one client at a time.

The IMAP server provides a superset of POP functionality but has a different interface. The IMAP server provides an off–line service, as well as an on–line service and a disconnected service. The IMAP protocol is designed to permit manipulation of remote mailboxes as if they were local. For example, clients can perform searches and mark messages with status flags such as ”deleted” or ”answered.” In addition, messages can remain in the server's database until explicitly removed. The IMAP server also allows simultaneous interactive access to user mailboxes by multiple clients.

Both the IMAP and POP servers are used for mail access only. These servers rely on the Simple Mail Transfer Protocol (SMTP) for sending mail.

Both IMAP and POP are open protocols, based on standards described in RFCs. The IMAP server is based on RFC 1730, and the POP server is based on RFC 1725. Both are connection–oriented using TCP sockets. The IMAP server listens on port 143, and the POP server listens on port 110. Both servers are handled by the **inetd** daemon.

Configuring IMAP and POP Servers

Prerequisites

You must have root authority.

Procedure

1. Uncomment the **imapd** and **pop3d** entries in the **/etc/inetd.conf** file.
2. Refresh the **inetd** daemon by running the following command:

   ```
   refresh –s inetd
   ```

Running Configuration Tests

Run a few tests to verify your **imapd** and **pop3d** servers are ready for operation.

First, verify the servers are listening on their ports. To do this, type the following commands at a command prompt, pressing the **Enter** key after each command:

```
netstat –a | grep imap
 netstat –a | grep pop
```

The following is the output from the **netstat** commands:

```
tcp     0       0     *.imap2         *.*        LISTEN
tcp     0       0     *.pop3          *.*        LISTEN
```

If you do not receive this output, recheck the entries in the **/etc/inetd.conf** file and rerun the **refresh –s inetd** command.

To test the configuration of the imapd server, telnet into the **imap2**, port `143`. When you connect via telnet, you will get the imapd prompt. You can then enter the IMAP Version 4 commands as defined in RFC 1730. To run one of the commands, type a period (.), followed by a space, and then the command name. For example:

```
.   CommandName
```

Note that passwords are echoed when you telnet into the **imapd** server.

In the following `telnet` example, you must provide your own password where *id_password* is indicated in the **login** command.

```
telnet e–xbelize 143
 Trying...
 Connected to e–xbelize.austin.ibm.com.
 Escape character is ’^]’.
 * OK e–xbelize.austin.ibm.com IMAP4 server ready
. login id   id_password
. OK
. examine /usr/spool/mail/root
 * FLAGS (\Answered \Flagged \Draft \Deleted \Seen)
 * OK [PERMANENTFLAGS (\Answered \Flagged \Draft \Deleted \Seen \*)]
 * 0 EXISTS
 * 0 RECENT
 * OK [UIDVALIDITY 823888143]
. OK [READ–ONLY] Examine completed
. logout
 * BYE Server terminating connection
. OK Logout completed
 Connection closed.
```

To test the configuration of the **pop3d** server, telnet into the Post Office Protocol Version 3 (POP3) port, `110`. When you telnet in, you should get the pop3d prompt. You can enter the POP commands that are defined in RFC 1725. To run one of the commands, type a period (.), followed by a space, and then the command name. For example:

```
.   CommandName
```

Note that passwords are echoed when you telnet into the **pop3d** server.

In the following `telnet` example, you must provide your own password where *id_password* is indicated in the **pass** command.

```
telnet e–xbelize 110
 Trying...
 Connected to e–xbelize.austin.ibm.com.
 Escape character is ’^]’.
 +OK e–xbelize.austin.ibm.com POP3 server ready
 user id
 +OK Name is a valid mailbox
 pass   id_password
 +OK Maildrop locked and ready
 list
 +OK scan listing follows
.
 stat
 +OK 0 0
 quit
 +OK
 Connection closed.
```

syslog Facility

The IMAP and POP server software sends log messages to the **syslog** facility.

To configure your system for IMAP and POP logging through the **syslog** facility, you must be the root user. Edit the **/etc/syslog.conf** file, and add an entry for ***.debug** as follows:

```
*.debug /usr/adm/imapd.log
```

The **`usr/adm/imapd.log`** file must exist before the **syslogd** daemon re–reads the **/etc/syslog.conf** configuration file. To create this file, type the following at a command line prompt and press **Enter**:

```
touch /usr/adm/imapd.log
```

Refresh the **syslogd** daemon to re–read its configuration file. Type the following at a command line prompt and press **Enter**:

```
refresh –s syslogd
```

Mail Reference

This section provides a quick reference to the various Mail commands, files, and directories.

List of Mail Commands

The following is a list of mail management commands.

bugfiler	Stores bug reports in specific mail directories.
comsat	Notifies users of incoming mail (daemon).
mailq	Prints the contents of the mail queue.
mailstats	Displays statistics about mail traffic.
newaliases	Builds a new copy of the alias database from the **/etc/mail/aliases** file.
rmail	Handles remote mail received through the **uucp** command of the Basic Networking Utilities (BNU).
sendbug	Mails a system bug report to a specific address.
sendmail	Routes mail for local or network delivery.
smdemon.cleanu	Cleans up the **sendmail** queue for periodic housekeeping.

List of Mail Files and Directories

This list of files and directories is arranged by function.

Note:: In versions earlier than AIX 5.1, the **sendmail.cf** and **aliases** files are located in **/etc/sendmail.cf** and **/etc/aliases**, respectively.

Using the Mail Program

/usr/share/lib/Mail.rc	Sets local system defaults for all users of the mail program. A text file you can modify to set the default characteristics of the **mail** command.
$HOME/.mailrc	Enables the user to change the local system defaults for the mail facility.
$HOME/mbox	Stores processed mail for the individual user.
/usr/bin/Mail, **/usr/bin/mail**, or **/usr/bin/mailx**	Specifies three names linked to the same program. The mail program is *one* of the user interfaces to the mail system.
/var/spool/mail	Specifies the default mail drop directory. By default, all mail is delivered to the **/var/spool/mail/UserName** file.
/usr/bin/bellmail	Performs local mail delivery.
/usr/bin/rmail	Performs remote mail interface for BNU.
/var/spool/mqueue	Contains the log file and temporary files associated with the messages in the mail queue.

Using the sendmail Command

/usr/sbin/sendmail
The **sendmail** command.

/usr/ucb/mailq
Links to the **/usr/sbin/sendmail**. Using **mailq** is equivalent to using the **/usr/sbin/sendmail –bp** command.

/usr/ucb/newaliases
Links to the **/usr/sbin/sendmail** file. Using **newaliases** is equivalent to using the **/usr/sbin/sendmail –bi** command.

/etc/netsvc.conf
Specifies the ordering of certain name resolution services.

/usr/sbin/mailstats
Formats and prints the **sendmail** statistics as found in the **/etc/sendmail.st** file if it exists. The **/etc/sendmail.st** file is the default, but you can specify an alternative file.

/etc/mail/aliases
Describes a text version of the aliases file for the **sendmail** command. You can edit this file to create, modify, or delete aliases for your system.

/etc/aliasesDB
Describes a directory containing the aliases database files, **DB.dir** and **DB.pag**, that are created from the **/etc/mail/aliases** file when you run the **sendmail –bi** command.

/etc/mail/sendmail.cf
Contains the **sendmail** configuration information in text form. Edit the file to change this information.

/usr/lib/smdemon.cleanu
Specifies a shell file that runs the mail queue and maintains the **sendmail** log files in the **/var/spool/mqueue** directory.

/etc/mail/statistics
Collects statistics about mail traffic. This file does not grow. Use the **/usr/sbin/mailstats** command to display the contents of this file. Delete this file if you do not want to collect this information.

/var/spool/mqueue
Describes a directory containing the temporary files associated with each message in the queue. The directory can contain the log file.

/var/spool/cron/crontabs
Describes a directory containing files that the **cron** daemon reads to determine which jobs to start. The **root** file contains a line to start the **smdemon.cleanu** shell script.

List of Internet Message Access Protocol and Post Office Protocol Commands

/usr/sbin/ imapd
The Internet Message Access Protocol (IMAP) server process.

/usr/sbin/ pop3d
The Post Office Protocol Version 3 (POP3) server process.

Chapter 4. Transmission Control Protocol/Internet Protocol

This chapter describes the Transmission Control Protocol/Internet Protocol (TCP/IP) suite of networking software. TCP/IP is a powerful and flexible industry–standard way of connecting multiple computers to other machines.

The topics discussed in this chapter are:

Note:: Most of the tasks discussed in this chapter require root authority.

Planning Your TCP/IP Network

Because TCP/IP is such a flexible networking tool, you can customize it to fit the specific needs of your organization. The following are the major issues you need to consider when planning your network. The details of these issues are discussed at length later. This list is intended only to introduce you to the issues.

1. Decide which type of network hardware you want to use: token–ring, Ethernet Version 2, IEEE 802.3, Fiber Distributed Data Interface (FDDI), Serial Optical Channel (SOC), or Serial Line Interface Protocol (SLIP).
2. Plan the physical layout of the network.

   Consider which functions each host machine will serve. For example, you must decide which machine or machines will serve as gateways before you cable the network.
3. Decide whether a *flat* network or a *hierarchical* network organization best fits your needs.

   If your network is fairly small, at a single site, and consists of one physical network, then a flat network probably suits your needs. If your network is very large or complex with multiple sites or multiple physical networks, a hierarchical network might be a more efficient network organization for you.
4. If your network is to be connected to other networks, you must plan how your gateways should be set up and configured. Things to consider are:
   a. Decide which machine or machines will serve as gateways.

   b. Decide whether you need to use static or dynamic routing, or a combination of the two. If you choose dynamic routing, decide which routing daemons each gateway will use in light of the types of communications protocols you need to support.
5. Decide on an addressing scheme.

   If your network will not be part of a larger internetwork, choose the addressing scheme that best fits your needs. If you want your network to be connected to a larger internetwork such as the Internet, you will have to obtain an official set of addresses from your internet service provider (ISP).
6. Decide whether your system needs to be divided into subnets. If so, decide how you will assign subnet masks.
7. Decide on a naming scheme. Each machine on the network needs its own unique host name.
8. Decide whether your network needs a name server for name resolution or if using the **/etc/hosts** file will be sufficient.

   If you choose to use name servers, consider the type of name servers you need and how many you need to serve your network efficiently.
9. Decide the types of services you want your network to provide to remote users; for example, mail services, print services, file sharing, remote login, remote command execution, and others.

Installation and Configuration for TCP/IP

For information on installing Transmission Control Protocol/Internet Protocol (TCP/IP), see the *AIX 5L Version 5.2 Installation Guide and Reference*.

Configuring TCP/IP

After the TCP/IP software is installed on your system, you are ready to begin configuring your system.

Many TCP/IP configuration tasks can be performed in more than one way, either by:

- Using the Web-based System Manager Network application (fast path **wsm network**)
- Using the System Management Interface Tool (SMIT)
- Editing a file format
- Issuing a command at the shell prompt.

For example, the **rc.net** shell script performs required minimum host configuration for TCP/IP during the system startup process (the **rc.net** script is run by the configuration manager program during the second boot phase). By using Web-based System Manager or SMIT to perform the host configuration, the **rc.net** file is configured automatically.

Alternatively, you can configure the **/etc/rc.bsdnet** file using a standard text editor. With this method, you can specify the traditional UNIX TCP/IP configuration commands such as **ifconfig**, **hostname**, and **route**. See List of TCP/IP Commands for further information. If using the file edit method, you must enter **smit configtcp** fast path and then select **BSD Style rc Configuration**.

A few tasks, such as configuring a name server, cannot be done using Web-based System Manager or SMIT.

Configuring Hosts

Each host machine on your network must be configured to function according to the needs of the end users and the network as a whole. For each host on the network, you must configure the network interface, set the Internet address, and set the host name. You also must set up static routes to gateways or other hosts, specify daemons to be started by default, and set up the **/etc/hosts** file for name resolution (or set up the host to use a name server for name resolution).

Configuring Hosts as Servers

If the host machine will have a specific function like serve as a gateway, file server, or name server, you must perform the necessary configuration tasks after the basic configuration is complete.

For example, if your network is organized hierarchically and you want to use the **Domain Name** protocol to resolve names into Internet addresses, you will need to configure at least one name server to provide this function for your network.

Remember, a server host does not have to be a dedicated machine, it can be used for other things as well. If the name server function for your network is fairly small, the machine might also be used as a workstation or as a file server for your network.

Note:: If your system has either NIS or NIS+ installed, these services can also provide name resolution. For more information, see *AIX 5L Version 5.2* Network Information Service *(NIS and NIS+) Guide*.

Configuring Gateways

If your network is going to communicate with other networks, you will need to configure at least one gateway host. You must consider which communications protocols you want to support, and then use whichever routing daemon (the **routed** or **gated** daemon) that supports those protocols.

TCP/IP System Manager Commands

The following list contains the commands used to configure and manage a TCP/IP network:

arp	Displays or changes the Internet address to hardware address translation tables used by the **Address Resolution** protocol.
finger	Returns information about users on a specified host.
host	Shows the Internet address of a specified host or the host name of a specified Internet address.
hostname	Shows or sets the Internet name and address of the local host.
ifconfig	Configures network interfaces and their characteristics.
netstat	Shows local and foreign addresses, routing tables, hardware statistics, and a summary of packets transferred.
no	Sets or shows current network kernel options.
ping	Determines whether a host is reachable.
route	Permits you to manipulate the routing tables manually.
ruptime	Shows status information on hosts that are connected to local physical networks and are running the **rwhod** server.
rwho	Shows status information for users on hosts that are connected to local physical networks and running the **rwhod** server.
setclock	Reads the network time service and sets the time and date of the local host accordingly.
timedc	Returns information about the **timed** daemon.
trpt	Reports protocol tracing on TCP sockets.
whois	Provides the Internet name directory service.

Configuring a TCP/IP Network Checklist

Use the following procedure as a guide for configuring your network. Ensure that you have read and understood the appropriate material.

After you bring your network up and it is running properly, you might find it useful to refer to this checklist for the purpose of debugging.

Prerequisites

1. Network hardware is installed and cabled. See TCP/IP Local Area Network Adapter Cards.
2. TCP/IP software is installed. See the *AIX 5L Version 5.2 Installation Guide and Reference*.

Procedure

1. Read TCP/IP Protocols on page 4-6 for the basic organization of TCP/IP. You should understand:

   – the layered nature of TCP/IP (that is, different protocols reside at different layers)

   – how data flows through the layers

2. Minimally configure each host machine on the network. This means adding a network adapter, assigning an IP address, and assigning a hostname to each host, as well as defining a default route to your network. Read TCP/IP Network Interfaces on page 4-50,

3. TCP/IP Addressing on page 4-58, and Choosing Names for the Hosts on Your Network on page 4-66.

   Note:: Each machine on the network needs this basic configuration whether it will be an end–user host, a file server, a gateway, or a name server.

4. Configure and start the **inetd** daemon on each host machine on the network. Read TCP/IP Daemons on page 4-174 and then follow the instructions in Configuring the inetd Daemon on page 4-177.

5. Configure each host machine to perform either local name resolution or to use a name server. If you are setting up a hierarchical Domain Name network, configure at least one host to function as a name server. Read and follow the instructions in TCP/IP Name Resolution on page 4-64.

6. If your network will communicate with any remote networks, configure at least one host to function as a gateway. The gateway can use static routes or a routing daemon to perform internetwork routing. Read and follow the instructions in TCP/IP Routing on page 4-180.

7. Decide which services each host machine on the network will use. By default, all services are available. Follow the instructions in Client Network Services on page 4-178 if you wish to make a particular service unavailable.

8. Decide which hosts on the network will be servers, and which services a particular server will provide. Follow the instructions in Server Network Services on page 4-179 to start the server daemons you wish to run.

9. Configure any remote print servers you will need. See Printer Overview in *AIX 5L Version 5.2 Guide to Printers and Printing* for more information.

10. If desired, configure a host to use or to serve as the master time server for the network. See the **timed** daemon in the *AIX 5L Version 5.2 Commands Reference* for more information.

TCP/IP Protocols

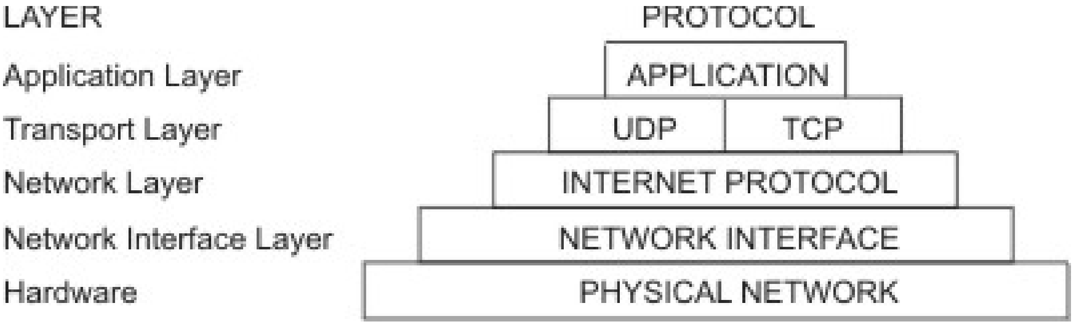
The topics discussed in this section are as follows:

- Internet Protocol (IP) Version 6 Overview on page 4-8
- Packet Tracing on page 4-16
- Network Interface Packet Headers on page 4-16
- Internet Network–Level Protocols on page 4-19
- Internet Transport–Level Protocols on page 4-25
- Internet Application–Level Protocols on page 4-29
- Assigned Numbers on page 4-34

Protocols are sets of rules for message formats and procedures that allow machines and application programs to exchange information. These rules must be followed by each machine involved in the communication in order for the receiving host to be able to understand the message.

The **TCP/IP** *suite* of protocols can be understood in terms of layers (or levels).

Figure 3. TCP/IP Suite of Protocols This illustration depicts the layers of the TCP/IP protocol. From the top they are, Application Layer, Transport Layer, Network Layer, Network Interface Layer, and Hardware.

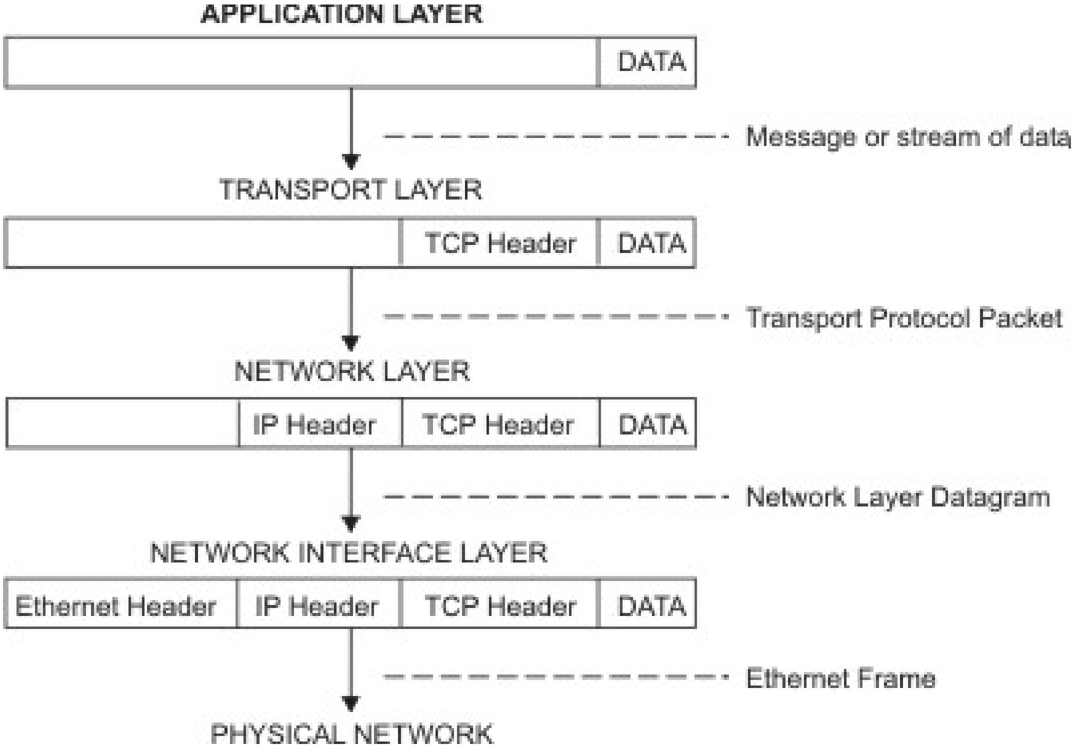


TCP/IP carefully defines how information moves from sender to receiver. First, application programs send messages or streams of data to one of the Internet Transport Layer Protocols, either the **User Datagram Protocol** (**UDP**) or the **Transmission Control Protocol** (**TCP**). These protocols receive the data from the application, divide it into smaller pieces called *packets*, add a destination address, and then pass the packets along to the next protocol layer, the Internet Network layer.

The Internet Network layer encloses the packet in an **Internet Protocol** (**IP**) datagram, puts in the datagram header and trailer, decides where to send the datagram (either directly to a destination or else to a gateway), and passes the datagram on to the Network Interface layer.

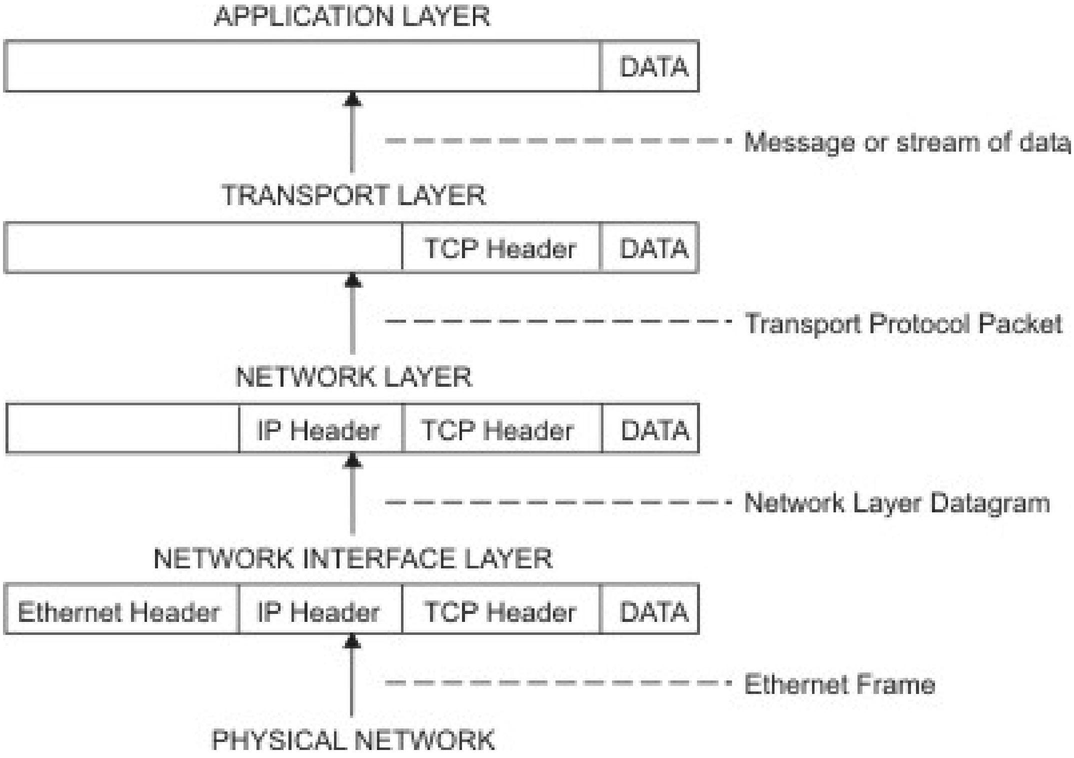
The Network Interface layer accepts **IP** datagrams and transmits them as *frames* over a specific network hardware, such as Ethernet or Token–Ring networks.

Figure 4. Movement of Information from Sender Application to Receiver Host This illustration shows the flow of information down the TCP/IP protocol layers from the Sender to the Host.



Frames received by a host go through the protocol layers in reverse. Each layer strips off the corresponding header information, until the data is back at the application layer.

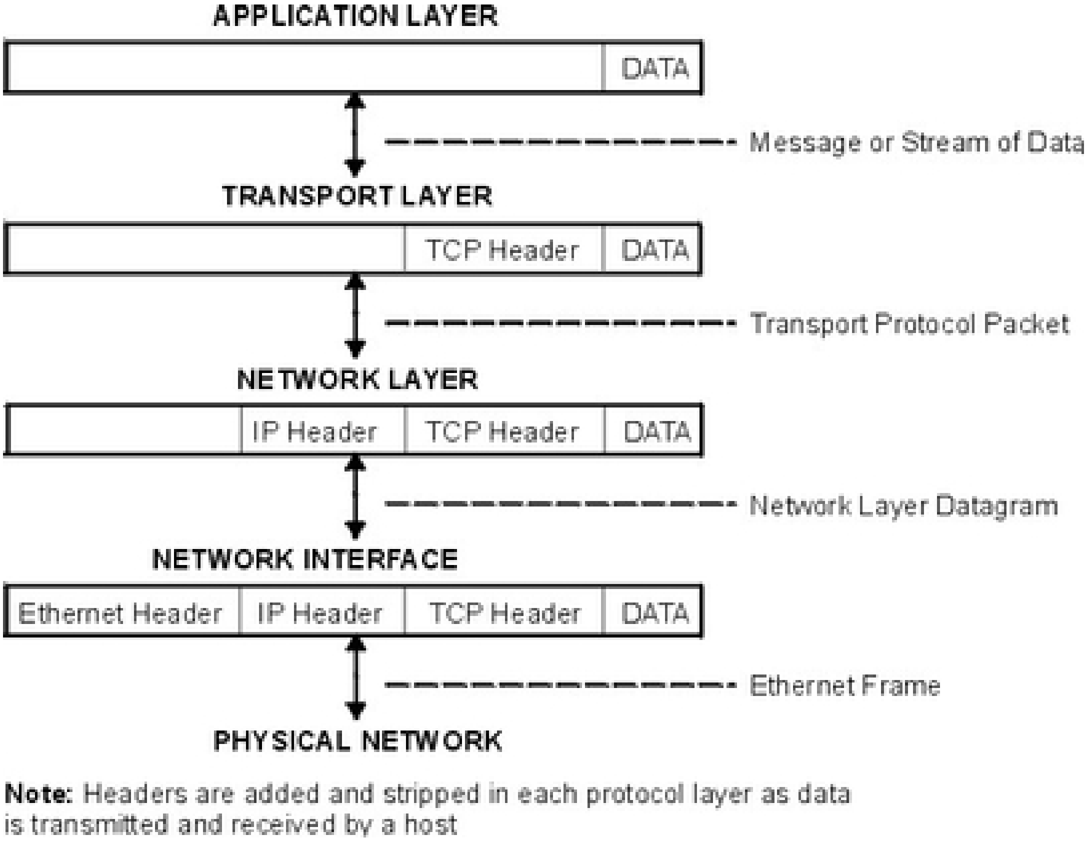
Figure 5. Movement of Information from Host to Application This illustration shows the flow of information up the TCP/IP protocol layers from the Host to the Sender.



Frames are received by the Network Interface layer (in this case, an Ethernet adapter). The Network Interface layer strips off the Ethernet header, and sends the datagram up to the Network layer. In the Network layer, the Internet Protocol strips off the IP header and sends the packet up to the Transport layer. In the Transport layer, the **TCP** (in this case) strips off the **TCP** header and sends the data up to the Application layer.

Hosts on a network send and receive information simultaneously. Figure 6 more accurately represents a host as it communicates.

Figure 6. Host Data Transmissions and Receptions This illustration shows data flowing both ways through the TCP/IP layers.



Internet Protocol (IP) Version 6 Overview

Internet Protocol (**IP**) Version 6 (IPv6 or IP *ng*) is the next generation of **IP** and has been designed to be an evolutionary step from **IP** Version 4 (IPv4). While IPv4 has allowed the development of a global Internet, it is not capable of carrying much farther into the future because of two fundamental factors: limited address space and routing complexity. The IPv4 32–bit addresses do not provide enough flexibility for global Internet routing. The deployment of Classless InterDomain Routing (CIDR) has extended the lifetime of IPv4 routing by a number of years, but the effort to better manage the routing will continue. Even if IPv4 routing could be scaled up, the Internet will eventually run out of network numbers.

The Internet Engineering Task Force (IETF) recognized that IPv4 would not be able to support the phenomenal growth of the Internet, so the IETF IP *ng* working group was formed. Of the proposals that were made, **Simple Internet Protocol Plus** (**SIPP**) was chosen as an evolutionary step in the development of IP. This was renamed to IP *ng*, and RFC1883 was finalized in December of 1995.

IPv6 extends the maximum number of Internet addresses to handle the ever increasing Internet user population. As an evolutionary change from IPv4, IPv6 has the advantage of allowing the new and the old to coexist on the same network. This coexistence enables an orderly migration from IPv4 (32 bit addressing) to IPv6 (128 bit addressing) on an operational network.

This overview is intended to give the reader a general understanding of the IP *ng* protocol. For detailed information, please see RFCs 2460, 2373, 2465, 1886, 2461, 2462, and 2553.

Expanded Routing and Addressing

IPv6 increases the **IP** address size from 32 bits to 128 bits, thereby supporting more levels of addressing hierarchy, a much greater number of addressable nodes, and simpler autoconfiguration of addresses.

IPv6 has three types of addresses:

unicast

A packet sent to a unicast address is delivered to the interface identified by that address. A unicast address has a particular scope: link–local, site–local, global. There are also two special unicast addresses:

- ::/128 (unspecified address)
- ::1/128 (loopback address)

multicast

A packet sent to a multicast address is delivered to all interfaces identified by that address. A multicast address is identified by the prefix ff::/8. As with unicast addresses, multicast addresses have a similar scope: node–local, link–local, site–local, and organization–local.

anycast

An anycast address is an address that has a single sender, multiple listeners, and only one responder (normally the ”nearest” one, according to the routing protocols’ measure of distance). For example, several web servers listening on an anycast address. When a request is sent to the anycast address, only one responds.

An anycast address is indistinguishable from a unicast address. A unicast address becomes an anycast address when more than one interface is configured with that address.

Note:: There are no broadcast addresses in IPv6. Their function has been superseded by the multicast address.

Autoconfiguration

The primary mechanisms available that enable a node to start up and communicate with other nodes over an IPv4 network are hard–coding, **BOOTP**, and **DHCP**.

IPv6 introduces the concept of *scope* to **IP** addresses, one of which is link–local. This allows a host to construct a valid address from the predefined link–local prefix and its local identifier. This local identifyer is typically derived from the medium access control (MAC) address of the interface to be configured. Using this address, the node can communicate with other hosts on the same subnet and, for a fully–isolated subnet, might not need any other address configuration.

Meaningful Addresses

With IPv4, the only generally recognizable meaning in addresses are broadcast (typically all 1s or all 0s), and classes (for example, a class D is multicast). With IPv6, the prefix can be quickly examined to determine *scope* (for example, link–local), multicast versus unicast, and a mechanism of assignment (provider–based or geography–based).

Routing information might be explicitly loaded into the upper bits of addresses as well, but this has not yet been finalized by the IETF (for provider–based addresses, routing information is implicitly present in the address).

Duplicate Address Detection

When an interface is initialized or reinitialized, it uses autoconfiguration to tentatively associate a link–local address with that interface (the address is not yet assigned to that interface in the traditional sense). At this point, the interface joins the all–nodes and solicited–nodes multicast groups, and sends a neighbor discovery message to these groups. By using the multicast address, the node can determine whether that particular link–local address has been previously assigned, and choose an alternate address. This eliminates accidentally assigning the same address to two different interfaces on the same link. (It is still possible to create duplicate global–scope addresses for nodes that are not on the same link.)

Neighbor Discovery/Stateless Address Autoconfiguration

Neighbor Discovery Protocol (**NDP**) for IPv6 is used by nodes (hosts and routers) to determine the link–layer addresses for neighbors known to reside on attached links, and maintain per–destination routing tables for active connections. Hosts also use **NDP** to find neighboring routers that are willing to forward packets on their behalf and detect changed link–layer addresses. **NDP** uses the **Internet Control Message Protocol** (**ICMP**) Version 6 with its own unique message types. In general terms, the IPv6 Neighbor Discovery protocol corresponds to a combination of the IPv4 **Address Resolution Protocol** (**ARP**), ICMP Router Discovery (RDISC), and **ICMP** Redirect (ICMPv4), but with many improvements over these IPv4 protocols.

IPv6 defines both a stateful and a stateless address autoconfiguration mechanism. *Stateless autoconfiguration* requires no manual configuration of hosts; minimal, if any, configuration of routers; and no additional servers. The stateless mechanism allows a host to generate its own addresses using a combination of locally available information and information advertised by routers. Routers advertise prefixes that identify the subnets associated with a link, while hosts generate an interface token that uniquely identifies an interface on a subnet. An address is formed by combining the two. In the absence of routers, a host can only generate link–local addresses. However, link–local addresses are sufficient for allowing communication among nodes attached to the same link.

Routing Simplification

To simplify routing issues, IPv6 addresses are considered in two parts: a prefix and an ID. This might seem the same as the IPv4 net–host address breakdown, but it has two advantages:

no class	No fixed number of bits for prefix or ID, which allows for a reduction in loss due to over–allocation
nesting	An arbitrary number of divisions can be employed by considering different numbers of bits as the prefix.

Case 1

128 bits
node address

Case 2

n bits	128– n bits
Subnet prefix	Interface ID

Case 3:

n bits	80– n bits	48 bits
Subscriber prefix	Subnet ID	Interface ID

Case 4:

s bits	n bits	m bits	128– s – n – m bits
Subscribe prefix	Area ID	Subnet ID	Interface ID

Generally, IPv4 cannot go beyond Case 3, even with Variable Length Subnet Mask (VLSM is a means of allocating **IP** addressing resources to subnets according to their individual need rather than some general network–wide rule). This is as much an artifact of the shorter address length as the definition of variable length prefixes, but is worth noting nonetheless.

Header Format Simplification

IPv6 simplifies the **IP** header, by removing entirely or by moving to an extension header, some of the fields found in the IPv4 header. It defines a more flexible format for optional information (the extension headers). Specifically, note the absence of:

- header length (length is constant)
- identification
- flags
- fragment offset (moved into fragmentation extension headers)
- header checksum (upper–layer protocol or security extension header handles data integrity).

Table 1. IPv4 header

Version	IHL	Type of Service	Total Length	
Identification			Flags	Fragment Offset
Time to Live		Protocol	Header Checksum	
Source Address				
Destination Address				
Options				Padding

Table 2. IPv6 header

<table>
<tr><td>Version</td><td>Prio</td><td colspan="3">Flow Label</td></tr>
<tr><td colspan="3">Payload Length</td><td>Next Header</td><td>Hop Limit</td></tr>
<tr><td colspan="5">Source Address</td></tr>
<tr><td colspan="5">Destination Address</td></tr>
</table>

IP *ng* includes an improved options mechanism over IPv4. IPv6 options are placed in separate extension headers that are located between the IPv6 header and the transport–layer header in a packet. Most extension headers are not examined or processed by any router along a packet delivery path until it arrives at its final destination. This mechanism facilitates a major improvement in router performance for packets containing options. In IPv4 the presence of any options requires the router to examine all options.

Another improvement is that, unlike IPv4 options, IPv6 extension headers can be of arbitrary length and the total amount of options carried in a packet is not limited to 40 bytes. This feature, plus the manner in which it is processed, permits IPv6 options to be used for functions that were not practical in IPv4, such as the IPv6 Authentication and Security Encapsulation options.

To improve the performance when handling subsequent option headers and the transport protocol that follows, IPv6 options are always an integer multiple of eight octets long to retain this alignment for subsequent headers.

By using extension headers instead of a protocol specifier and options fields, newly defined extensions can be integrated more easily.

Current specifications define extension headers in the following ways:

- Hop–by–hop options that apply to each hop (router) along the path
- Routing header for loose/strict source routing (used infrequently)
- A fragment defines the packet as a fragment and contains information about the fragment (IPv6 routers do not fragment)
- Authentication (See IP Security in *AIX 5L Version 5.2 Security Guide*
- Encryption (See IP Security in *AIX 5L Version 5.2 Security Guide*
- Destination options for the destination node (ignored by routers).

Improved Quality–of–Service/Traffic Control

While quality of service can be controlled by use of a control protocol such as **RSVP**, IPv6 provides for explicit priority definition for packets by using the priority field in the **IP** header. A node can set this value to indicate the relative priority of a particular packet or set of packets, which can then be used by the node, one or more routers, or the destination to make choices concerning the packet (that is, dropping it or not).

IPv6 specifies two types of priorities, those for congestion–controlled traffic, and those for non–congestion–controlled traffic. No relative ordering is implied between the two types.

Congestion–controlled traffic is defined as traffic that responds to congestion through some sort of ”back–off” or other limiting algorithm. Priorities for congestion–controlled traffic are:

0	uncharacterized traffic
1	”filler” traffic (for example, netnews)
2	unattended data transfer (for example, mail)
3	(reserved)
4	attended bulk transfer (for example, FTP)
5	(reserved)
6	interactive traffic (for example, Telnet)
7	control traffic (for example, routing protocols)

Non–congestion–controlled traffic is defined as traffic that responds to congestion by dropping (or simply not resending) packets, such as video, audio, or other real–time traffic. Explicit levels are not defined with examples, but the ordering is similar to that for congestion–controlled traffic:

- The lowest value that the source is most willing to have discarded should be used for traffic.
- The highest value that the source is least willing to have discarded should be used for traffic.

This priority control is only applicable to traffic from a particular source address. Control traffic from one address is not an explicitly higher priority than attended bulk transfer from another address.

Flow Labeling

Outside of basic prioritization of traffic, IPv6 defines a mechanism for specifying a particular flow of packets. In IPv6 terms, a *flow* is defined as a sequence of packets sent from a particular source to a particular (unicast or multicast) destination for which the source desires special handling by the intervening routers.

This flow identification can be used for priority control, but might also be used for any number of other controls.

The flow label is chosen randomly, and does not identify any characteristic of the traffic other than the flow to which it belongs. This means that a router cannot determine that a packet is a particular type by examining the flow label. It can, however, determine that it is part of the same sequence of packets as the last packet containing that label.

Note:: Until IPv6 is in general use, the flow label is mostly experimental. Uses and controls involving flow labels have not yet been defined nor standardized.

Tunneling

The key to a successful IPv6 transition is compatibility with the existing installed base of IPv4 hosts and routers. Maintaining compatibility with IPv4 while deploying IPv6 streamlines the task of transitioning the Internet to IPv6.

While the IPv6 infrastructure is being deployed, the existing IPv4 routing infrastructure can remain functional, and can be used to carry IPv6 traffic. Tunneling provides a way to use an existing IPv4 routing infrastructure to carry IPv6 traffic.

IPv6 or IPv4 hosts and routers can tunnel IPv6 datagrams over regions of IPv4 routing topology by encapsulating them within IPv4 packets. Tunneling can be used in a variety of ways:

Router–to–Router	IPv6 or IPv4 routers interconnected by an IPv4 infrastructure can tunnel IPv6 packets between themselves. In this case, the tunnel spans one segment of the end–to–end path that the IPv6 packet takes.
Host–to–Router	IPv6 or IPv4 hosts can tunnel IPv6 packets to an intermediary IPv6 or IPv4 router that is reachable through an IPv4 infrastructure. This type of tunnel spans the first segment of the packet's end–to–end path.
Host–to–Host	IPv6 or IPv4 hosts that are interconnected by an IPv4 infrastructure can tunnel IPv6 packets between themselves. In this case, the tunnel spans the entire end–to–end path that the packet takes.
Router–to–Host	IPv6/IPv4 routers can tunnel IPv6 packets to their final destination IPv6 or IPv4 host. This tunnel spans only the last segment of the end–to–end path.

Tunneling techniques are usually classified according to the mechanism by which the encapsulating node determines the address of the node at the end of the tunnel. In router–to–router or host–to–router methods, the IPv6 packet is being tunneled to a router. In host–to–host or router–to–host methods, the IPv6 packet is tunneled all the way to its final destination.

The entry node of the tunnel (the encapsulating node) creates an encapsulating IPv4 header and transmits the encapsulated packet. The exit node of the tunnel (the decapsulating node) receives the encapsulated packet, removes the IPv4 header, updates the IPv6 header, and processes the received IPv6 packet. However, the encapsulating node needs to maintain soft state information for each tunnel, such as the maximum transmission unit (MTU) of the tunnel, to process IPv6 packets forwarded into the tunnel.

IPv6 Security

For details about IP Security, versions 4 and 6, see IP Security in *AIX 5L Version 5.2 Security Guide*.

IPv6 Multihomed Link–Local and Site–Local Support

A host can have more than one interface defined. A host with two or more active interfaces is called multihomed. Each interface has a link–local address associated with it. Link–local addresses are sufficient for allowing communication among nodes attached to the same link.

A multihomed host has two or more associated link–local addresses. The AIX IPv6 implementation has 4 options to handle how link–layer address resolution is resolved on multihomed hosts. Option 1 is the default.

Option 0
No multihomed actions are taken. Transmissions will go out on the first link–local interface. When the **Neighbor Discovery Protocol** (**NDP**) must perform address resolution, it multicasts a Neighbor Solicitation message out on each interface with a link local address defined. NDP queues the data packet until the first Neighbor Advertisement message is received. The data packet is then sent out on this link.

Option 1
When the **NDP** must perform address resolution, that is, when sending a data packet to a destination and the the link–layer information for the next hop is not in the Neighbor Cache, it multicasts a Neighbor Solicitation message out on each interface with a link–local address defined. **NDP** then queues the data packet until it gets the link–layer information. **NDP** then waits until a response is received for each interface. This guarantees that the data packets are sent on the appropriate outgoing interfaces. If **NDP** did not wait, but responded to the first Neighbor Advertisement received, it would be possible for a data packet to be sent out on a link not associated with the packet source address. Because **NDP** must wait, a delay in the first packet being sent occurs. However, the delay occurrs anyway in waiting for the first response.

Option 2
Multihomed operation is allowed, but dispatching of a data packet is limited to the interface specified by main_if6. When the **NDP** must perform address resolution, it multicasts a Neighbor Solicitation message out on each interface with a link–local address defined. It then waits for a Neighbor Advertisement message from the interface specified by main_if6 (see the **no** command). Upon receiving a response from this interface, the data packet is sent out on this link.

Option 3
Multihomed operation is allowed, but dispatching of a data packet is limited to the interface specified by main_if6 and site–local addresses are only routed for the interface specified by main_site6 (see the **no** command). The NDP operates just as it does for Option 2. For applications that route data packets using site–local addresses on a multihomed host, only the site–local address specified by main_site6 are used.

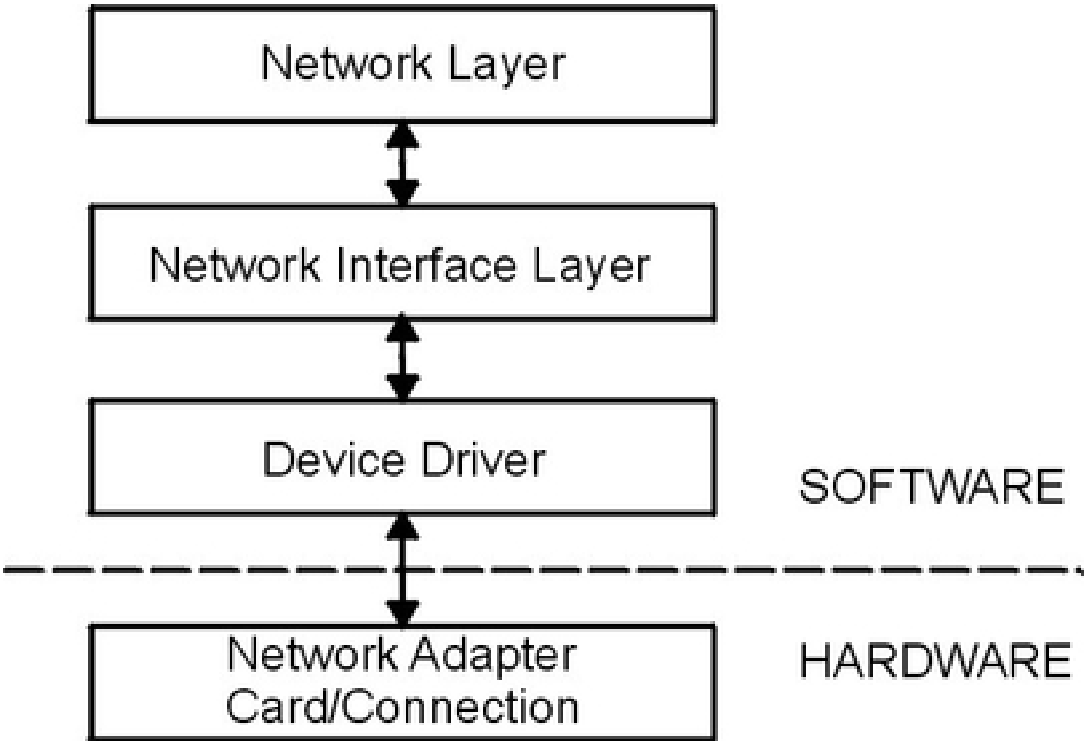
Packet Tracing

Packet tracing is the process by which you can verify the path of a packet through the layers to its destination. The **iptrace** command performs network interface level packet tracing. The **ipreport** command issues output on the packet trace in both hexadecimal and ASCII format. The **trpt** command performs transport protocol level packet tracking for the **TCP**. The **trpt** command output is more detailed, including information on time, **TCP** state, and packet sequencing.

Network Interface Packet Headers

At the Network Interface layer, packet headers are attached to outgoing data.

Figure 7. Packet Flow through Network Interface Structure This illustration shows bi–directional data flow through the layers of the Network Interface Structure. From the top (software) they are the Network Layer, Network Interface Layer, Device Driver, and the (hardware) Network Adapter Card or Connection.



Packets are then sent through the network adapter to the appropriate network. Packets can pass through many gateways before reaching their destinations. At the destination network, the headers are stripped from the packets and the data is sent to the appropriate host.

The following section contains packet header information for several of the more common network interfaces.

Ethernet Adapter Frame Headers

An **Internet Protocol** (**IP**) or **Address Resolution Protocol** (**ARP**) frame header for the Ethernet adapter is composed of three fields as shown in the following table..

Ethernet adapter frame header

Field	**Length**	**Definition**
DA	6 bytes	Destination address.
SA	6 bytes	Source address. If bit 0 of this field is set to 1, it indicates that routing information (RI) is present.
Type	2 bytes	Specifies whether the packet is **IP** or **ARP**. The type number values are listed below.

Type field numbers:

IP	0800
ARP	0806

Token–Ring Frame Headers

The medium access control (MAC) header for the token–ring adapter is composed of five fields, as shown in the following table.

Token–ring MAC header

Field	**Length**	**Definition**
AC	1 byte	Access control. The value in this field x‘00’ gives the header priority 0.
FC	1 byte	Field control. The value in this field x‘40’ specifies the Logical Link Control frame.
DA	6 bytes	Destination address.
SA	6 bytes	Source address. If bit 0 of this field is set to 1, it indicates that routing information (RI) is present.
RI	18 bytes	Routing information. The valid fields are discussed below.

The MAC header consists of two routing information fields of two bytes each: routing control (RC) and segment numbers. A maximum of eight segment numbers can be used to specify recipients of a limited broadcast. RC information is contained in bytes 0 and 1 of the RI field. The settings of the first two bits of the RC field have the following meanings:

bit (0) = 0	Use the nonbroadcast route specified in the RI field.
bit (0) = 1	Create the RI field and broadcast to all rings.
bit (1) = 0	Broadcast through all bridges.
bit (1) = 1	Broadcast through limited bridges.

The logical link control (LLC) header is composed of five fields, as shown in the following LLC header table.

802.3 LLC header

Field	**Length**	**Definition**
DSAP	1 byte	Destination service access point. The value in this field is x‘aa’.
SSAP	1 byte	Source service access point. The value in this field is x‘aa’.
CONTROL	1 byte	Determines the LLC commands and responses. The three possible values for this field are discussed below.
PROT_ID	3 bytes	Protocol ID. This field is reserved. It has a value of x‘0’.
TYPE	2 bytes	Specifies whether the packet is **IP** or **ARP**.

Control Field Values

x‘03’
Unnumbered Information (UI) frame. This is the normal, or unsequenced, way in which token–ring adapter data is transmitted through the network. **TCP/IP** sequences the data.

x‘AF’
Exchange identification (XID) frame. This frame conveys the characteristics of the sending host.

x‘E3’
Test frame. This frame supports testing of the transmission path, echoing back the data that is received.

802.3 Frame Headers

The MAC header for the 802.3 adapter is composed of two fields, as shown in the following MAC header table.

802.3 MAC header

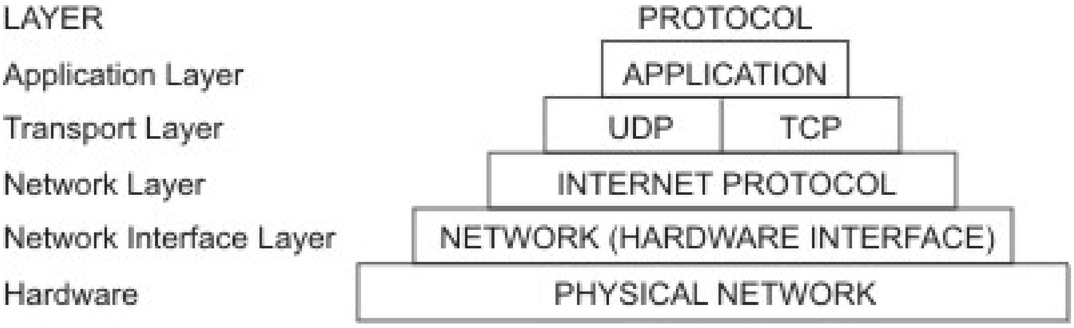
Field	Length	Definition
DA	6 bytes	Destination address.
SA	6 bytes	Source address. If bit 0 of this field is set to 1, it indicates that routing information (RI) is present.

The LLC header for 802.3 is the same as for Token–Ring MAC header.

Internet Network–Level Protocols

The Internet network–level protocols handle machine–to–machine communication. In other words, this layer implements **TCP/IP** routing. These protocols accept requests to send packets (along with the network address of the destination machine) from the Transport layer, convert the packets to datagram format, and send them down to the Network Interface layer for further processing.

Figure 8. Network Layer of the TCP/IP Suite of Protocols This illustration shows the various layers of the TCP/IP Suite of Protocols. From the top, the application layer consists of the application. The transport layer contains UDP and TCP. The network layer contains the network (hardware) interface. And finally, the hardware layer contains the physical network.



TCP/IP provides the protocols that are required to comply with RFC 1100, *Official Internet Protocols*, as well as other protocols commonly used by hosts in the Internet community.

Note:: The use of Internet network, version, socket, service, and protocol numbers in **TCP/IP** also complies with RFC 1010, *Assigned Numbers*.

Address Resolution Protocol

The first network–level protocol is the **Address Resolution Protocol** (**ARP**). **ARP** dynamically translates Internet addresses into the unique hardware addresses on local area networks.

To illustrate how **ARP** works, consider two nodes, X and Y. If node X wishes to communicate with Y, and X and Y are on different local area networks (LANs), X and Y communicate through *bridges*, *routers*, or *gateways*, using IP addresses. Within a LAN, nodes communicate using low–level hardware addresses.

Nodes on the same segment of the same LAN use **ARP** to determine the hardware address of other nodes. First, node X broadcasts an **ARP** request for node Y’s hardware address.

The **ARP** request contains X 's **IP** and hardware addresses, and Y 's **IP** address. When Y receives the **ARP** request, it places an entry for X in its **ARP** cache (which is used to map quickly from IP address to hardware address), then responds directly to X with an **ARP** response containing Y 's **IP** and hardware addresses. When node X receives Y 's **ARP** response, it places an entry for Y in its **ARP** cache.

Once an **ARP** cache entry exists at X for Y, node X is able to send packets directly to Y without resorting again to **ARP** (unless the **ARP** cache entry for Y is deleted, in which case **ARP** is reused to contact Y).

Unlike most protocols, **ARP** packets do not have fixed–format headers. Instead, the message is designed to be useful with a variety of network technologies, such as:

- Ethernet LAN adapter (supports both Ethernet and 802.3 protocols)
- Token–ring network adapter
- Fiber Distributed Data Interface (FDDI) network adapter

However, ARP does not translate addresses for **Serial Line Interface Protocol** (**SLIP**) or **Serial Optical Channel Converter** (**SOC**), since these are point–to–point connections.

The kernel maintains the translation tables, and the **ARP** is not directly available to users or applications. When an application sends an Internet packet to one of the interface drivers, the driver requests the appropriate address mapping. If the mapping is not in the table, an **ARP** broadcast packet is sent through the requesting interface driver to the hosts on the local area network.

Entries in the **ARP** mapping table are deleted after 20 minutes; incomplete entries are deleted after 3 minutes. To make a permanent entry in the **ARP** mapping tables, use the **arp** command with the *pub* parameter:

```
arp –s 802.3 host2 0:dd:0:a:8s:0 pub
```

When any host that supports **ARP** receives an **ARP** request packet, the host notes the **IP** and hardware addresses of the requesting system and updates its mapping table, if necessary. If the receiving host **IP** address does not match the requested address, the host discards the request packet. If the **IP** address does match, the receiving host sends a response packet to the requesting system. The requesting system stores the new mapping and uses it to transmit any similar pending Internet packets.

Internet Control Message Protocol

The second network–level protocol is the **Internet Control Message Protocol** (**ICMP**). **ICMP** is a required part of every **IP** implementation. **ICMP** handles error and control messages for **IP**. This protocol allows gateways and hosts to send problem reports to the machine sending a packet. **ICMP** does the following:

- Tests whether a destination is alive and reachable
- Reports parameter problems with a datagram header
- Performs clock synchronization and transit time estimations
- Obtains Internet addresses and subnet masks

Note:: **ICMP** uses the basic support of **IP** as if it were a higher–level protocol. However, **ICMP** is actually an integral part of **IP** and must be implemented by every **IP** module.

ICMP provides feedback about problems in the communications environment, but does not make **IP** reliable. That is, **ICMP** does not guarantee that an **IP** packet is delivered reliably or that an **ICMP** message is returned to the source host when an **IP** packet is not delivered or is incorrectly delivered.

ICMP messages might be sent in any of the following situations:

- When a packet cannot reach its destination
- When a gateway host does not have the buffering capacity to forward a packet
- When a gateway can direct a host to send traffic on a shorter route

TCP/IP sends and receives several ICMP message types (see Internet Control Message Protocol Message Types on page 4-21). **ICMP** is embedded in the kernel, and no application programming interface (API) is provided to this protocol.

Internet Control Message Protocol Message Types

ICMP sends and receives the following message types:

echo request Sent by hosts and gateways to test whether a destination is alive and reachable.

information request Sent by hosts and gateways to obtain an Internet address for a network to which they are attached. This message type is sent with the network portion of **IP** destination address set to a value of 0.

timestamp request Sent to request that the destination machine return its current value for time of day.

address mask request Sent by host to learn its subnet mask. The host can either send to a gateway, if it knows the gateway address, or send a broadcast message.

destination unreachable Sent when a gateway cannot deliver an **IP** datagram.

source quench Sent by discarding machine when datagrams arrive too quickly for a gateway or host to process, in order to request that the original source slow down its rate of sending datagrams.

redirect message Sent when a gateway detects that some host is using a nonoptimum route.

echo reply Sent by any machine that receives an echo request in reply to the machine which sent the request.

information reply Sent by gateways in response to requests for network addresses, with both the source and destination fields of the **IP** datagram specified.

timestamp reply Sent with current value of time of day.

address mask reply Sent to machines requesting subnet masks.

parameter problem Sent when a host or gateway finds a problem with a datagram header.

time exceeded Sent when the following are true:

- Each **IP** datagram contains a time–to–live counter (hop count), which is decremented by each gateway.
- A gateway discards a datagram because its hop count has reached a value of 0.

Internet Timestamp Used to record the time stamps through the route.

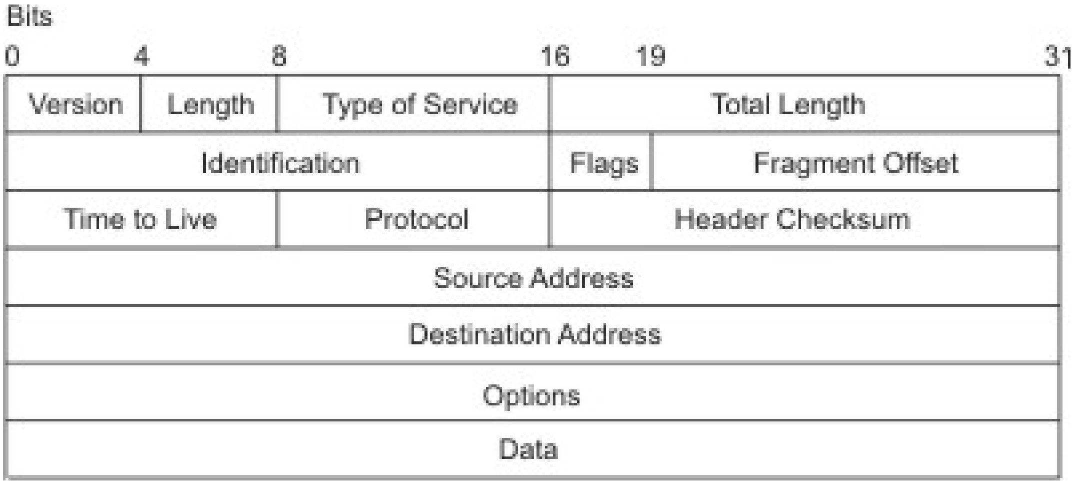
Internet Protocol

The third network–level protocol is the **Internet Protocol** (**IP**), which provides unreliable, connectionless packet delivery for the Internet. **IP** is connectionless because it treats each packet of information independently. It is unreliable because it does not guarantee delivery, meaning, it does not require acknowledgments from the sending host, the receiving host, or intermediate hosts.

IP provides the interface to the network interface level protocols. The physical connections of a network transfer information in a frame with a header and data. The header contains the source address and the destination address. **IP** uses an Internet datagram that contains information similar to the physical frame. The datagram also has a header containing Internet addresses of both source and destination of the data.

IP defines the format of all the data sent over the Internet.

Figure 9. Internet Protocol Packet Header This illustration shows the first 32 bits of a typical IP packet header. Table below lists the various entities.



Bits

0	4	8	16	19 31
Version	Length	Type of Service	Total Length	
Identification			Flags	Fragment Offset
Time to Live		Protocol	Header Checksum	
Source Address				
Destination Address				
Options				
Data				

IP Header Field Definitions

Version
Specifies the version of the **IP** used. The current version of the **IP** protocol is 4.

Length
Specifies the datagram header length, measured in 32–bit words.

Type of Service
Contains five subfields that specify the type of precedence, delay, throughput, and reliability desired for that packet. (The Internet does not guarantee this request.) The default settings for these five subfields are routine precedence, normal delay, normal throughput, and normal reliability. This field is not generally used by the Internet at this time. This implementation of **IP** complies with the requirements of the **IP** specification, RFC 791, *Internet Protocol*.

Total Length
Specifies the length of the datagram including both the header and the data measured in octets. Packet fragmentation at gateways, with reassembly at destinations, is provided. The total length of the **IP** packet can be configured on an interface–by–interface basis with the Web–based System Manager, **wsm**, the **ifconfig** command, or the System Management Interface Tool (SMIT) fast path, **smit chinet**. Use Web-based System Manager or SMIT to set the values permanently in the configuration database; use the **ifconfig** command to set or change the values in the running system.

Identification
Contains a unique integer that identifies the datagram.

Flags
Controls datagram fragmentation, along with the Identification field. The Fragment Flags specify whether the datagram can be fragmented and whether the current fragment is the last one.

Fragment Offset
Specifies the offset of this fragment in the original datagram measured in units of 8 octets.

Time to Live
Specifies how long the datagram can remain on the Internet. This keeps misrouted datagrams from remaining on the Internet indefinitely. The default time to live is 255 seconds.

Protocol
Specifies the high–level protocol type.

Header Checksum
Indicates a number computed to ensure the integrity of header values.

Source Address
Specifies the Internet address of the sending host.

Destination Address
Specifies the Internet address of the receiving host.

Options Provides network testing and debugging. This field is not required for every datagram.

End of Option List
Indicates the end of the option list. It is used at the end of the final option, not at the end of each option individually. This option should be used only if the end of the options would not otherwise coincide with the end of the **IP** header. End of Option List is used if options exceed the length of the datagram.

No Operation
Provides alignment between other options; for example, to align the beginning of a subsequent option on a 32–bit boundary.

Loose Source and Record Route
Provides a means for the source of an Internet datagram to supply routing information used by the gateways in forwarding the datagram to a destination and in recording the route information. This is a *loose* source route: the gateway or host **IP** is allowed to use any route of any number of other intermediate gateways in order to reach the next address in the route.

Strict Source and Record Route
Provides a means for the source of an Internet datagram to supply routing information used by the gateways in forwarding the datagram to a destination and in recording the route information. This is a *strict* source route: In order to reach the next gateway or host specified in the route, the gateway or host **IP** must send the datagram directly to the next address in the source route and only to the directly connected network that is indicated in the next address.

Record Route
Provides a means to record the route of an Internet datagram.

Stream Identifier
Provides a way for a stream identifier to be carried through networks that do not support the stream concept.

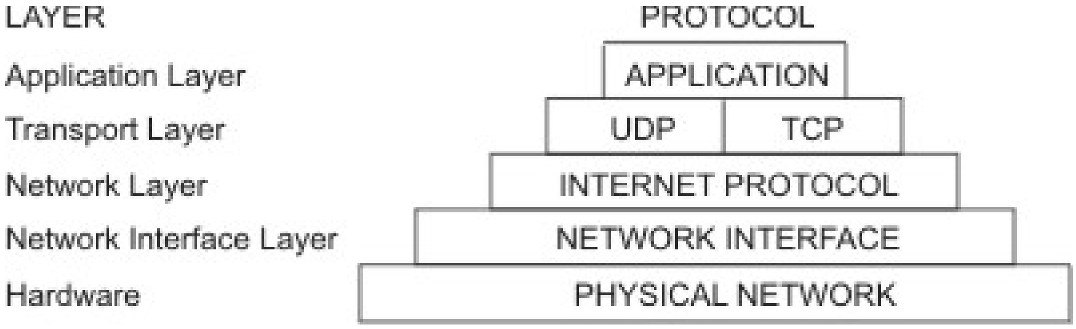
Internet Timestamp
Provides a record of the time stamps through the route.

Outgoing packets automatically have an **IP** header prefixed to them. Incoming packets have their **IP** header removed before being sent to the higher–level protocols. The **IP** protocol provides for the universal addressing of hosts in the Internet network.

Internet Transport–Level Protocols

The **TCP/IP** transport–level protocols allow application programs to communicate with other application programs.

Figure 10. Transport Layer of the TCP/IP Suite of Protocols This illustration shows the various layers of the TCP/IP Suite of Protocols. From the top, the application layer consists of the application. The transport layer contains UDP and TCP. The network layer contains the network (hardware) interface. And finally, the hardware layer contains the physical network.



User Datagram Protocol (**UDP**) and the **TCP** are the basic transport–level protocols for making connections between Internet hosts. Both **TCP** and **UDP** allow programs to send messages to and receive messages from applications on other hosts. When an application sends a request to the Transport layer to send a message, **UDP** and **TCP** break the information into packets, add a packet header including the destination address, and send the information to the Network layer for further processing. Both **TCP** and **UDP** use protocol ports on the host to identify the specific destination of the message.

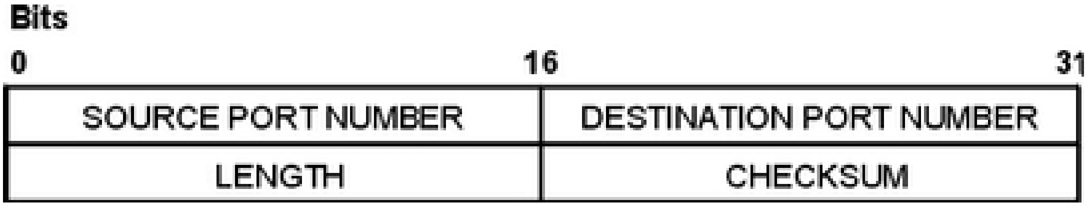
Higher–level protocols and applications use **UDP** to make datagram connections and **TCP** to make stream connections. The operating system sockets interface implements these protocols.

User Datagram Protocol

Sometimes an application on a network needs to send messages to a specific application or process on another network. The **UDP** provides a datagram means of communication between applications on Internet hosts. Because senders do not know which processes are active at any given moment, **UDP** uses destination protocol ports (or abstract destination points within a machine), identified by positive integers, to send messages to one of multiple destinations on a host. The protocol ports receive and hold messages in queues until applications on the receiving network can retrieve them.

Since **UDP** relies on the underlying **IP** to send its datagrams, **UDP** provides the same connectionless message delivery as **IP**. It offers no assurance of datagram delivery or duplication protection. However, **UDP** does allow the sender to specify source and destination port numbers for the message and calculates a checksum of both the data and header. These two features allow the sending and receiving applications to ensure the correct delivery of a message.

Figure 11. User Datagram Protocol (UDP) Packet Header This illustration shows the first 32 bits of the UDP packet header. The first 16 bits contain the source port number and the length. The second 16 bits contain the destination port number and the checksum.



SOURCE PORT NUMBER	DESTINATION PORT NUMBER
LENGTH	CHECKSUM

Applications that require reliable delivery of datagrams must implement their own reliability checks when using **UDP**. Applications that require reliable delivery of streams of data should use **TCP**.

UDP Header Field Definitions

Source Port Number Address of the protocol port sending the information.

Destination Port Number Address of the protocol port receiving the information.

Length Length in octets of the **UDP** datagram.

Checksum Provides a check on the **UDP** datagram using the same algorithm as the **IP**.

The applications programming interface (API) to **UDP** is a set of library subroutines provided by the sockets interface.

Transmission Control Protocol

TCP provides reliable stream delivery of data between Internet hosts. Like **UPD**, **TCP** uses Internet Protocol, the underlying protocol, to transport datagrams, and supports the block transmission of a continuous stream of datagrams between process ports. Unlike **UDP**, **TCP** provides reliable message delivery. **TCP** ensures that data is not damaged, lost, duplicated, or delivered out of order to a receiving process. This assurance of transport reliability keeps applications programmers from having to build communications safeguards into their software.

The following are operational characteristics of **TCP**:

Basic Data Transfer **TCP** can transfer a continuous stream of 8–bit octets in each direction between its users by packaging some number of bytes into segments for transmission through the Internet system. **TCP** implementation allows a segment size of at least 1024 bytes. In general, **TCP** decides when to block and forward packets at its own convenience.

Reliability **TCP** must recover data that is damaged, lost, duplicated, or delivered out of order by the Internet. **TCP** achieves this reliability by assigning a sequence number to each octet it transmits and requiring a positive acknowledgment (ACK) from the receiving **TCP**. If the ACK is not received within the time–out interval, the data is retransmitted. The **TCP** retransmission time–out value is dynamically determined for each connection, based on round–trip time. At the receiver, the sequence numbers are used to correctly order segments that may be received out of order and to eliminate duplicates. Damage is handled by adding a checksum to each segment transmitted, checking it at the receiver, and discarding damaged segments.

Flow Control **TCP** governs the amount of data sent by returning a window with every ACK to indicate a range of acceptable sequence numbers beyond the last segment successfully received. The window indicates an allowed number of octets that the sender may transmit before receiving further permission.

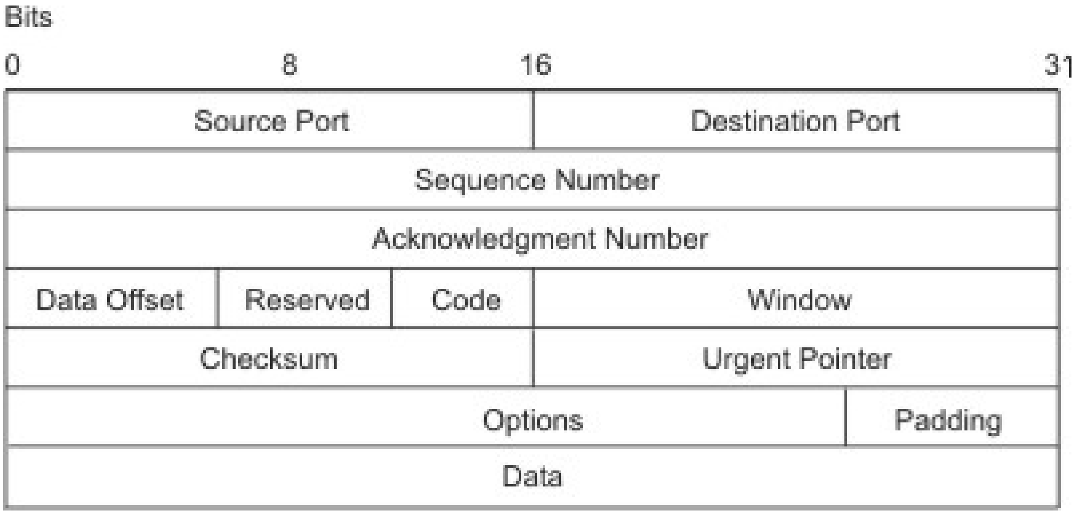
Multiplexing **TCP** allows many processes within a single host to use **TCP** communications facilities simultaneously. **TCP** receives a set of addresses of ports within each host. **TCP** combines the port number with the network address and the host address to uniquely identify each socket. A pair of sockets uniquely identifies each connection.

Connections **TCP** must initialize and maintain certain status information for each data stream. The combination of this information, including sockets, sequence numbers, and window sizes, is called a connection. Each connection is uniquely specified by a pair of sockets identifying its two sides.

Precedence and Security Users of **TCP** may indicate the security and precedence of their communications. Default values are used when these features are not needed.

The **TCP Packet Header** figure illustrates these characteristics.

Figure 12. Transmission Control Protocol (TCP) Packet Header This illustration shows what is contained in the TCP packet header. The individual entities are listed in the text below.



TCP Header Field Definitions

Source Port Identifies the port number of a source application program.

Destination Port Identifies the port number of a destination application program.

Sequence Number Specifies the sequence number of the first byte of data in this segment.

Acknowledgment Number Identifies the position of the highest byte received.

Data Offset Specifies the offset of data portion of the segment.

Reserved Reserved for future use.

Code Control bits to identify the purpose of the segment:

URG
Urgent pointer field is valid.

ACK
Acknowledgement field is valid.

PSH
Segment requests a PUSH.

RTS
Resets the connection.

SYN
Synchronizes the sequence numbers.

FIN
Sender has reached the end of its byte stream.

Window Specifies the amount of data the destination is willing to accept.

Checksum Verifies the integrity of the segment header and data.

Urgent Pointer Indicates data that is to be delivered as quickly as possible. This pointer specifies the position where urgent data ends.

Options

End of Option List
Indicates the end of the option list. It is used at the final option, not at the end of each option individually. This option needs to be used only if the end of the options would not otherwise coincide with the end of the **TCP** header.

No Operation
Indicates boundaries between options. Can be used between other options; for example, to align the beginning of a subsequent option on a word boundary. There is no guarantee that senders will use this option, so receivers must be prepared to process options even if they do not begin on a word boundary.

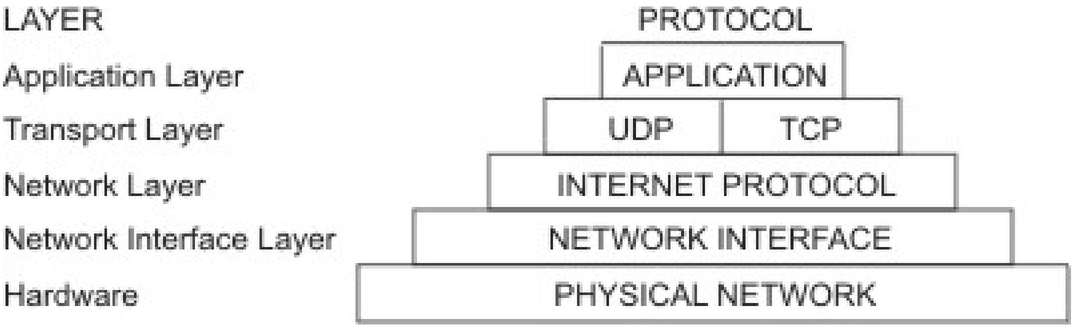
Maximum Segment Size
Indicates the maximum segment size **TCP** can receive. This is only sent in the initial connection request.

The applications programming interface to **TCP** consists of a set of library subroutines provided by the sockets interface.

Internet Application–Level Protocols

TCP/IP implements higher–level Internet protocols at the application program level.

Figure 13. Applicaton Layer of the TCP/IP Suite of Protocols This illustration shows the various layers of the TCP/IP Suite of Protocols. From the top, the application layer consists of the application. The transport layer contains UDP and TCP. The network layer contains the network (hardware) interface. And finally, the hardware layer contains the physical network.



When an application needs to send data to another application on another host, the applications send the information down to the transport level protocols to prepare the information for transmission.

The official Internet application–level protocols include:

- **Domain Name Protocol** (Domain Name Protocol)
- **Exterior Gateway Protoco** l (Exterior Gateway Protocol)
- **File Transfer Protocol** (File Transfer Protocol)
- **Name/Finger Protocol** (Name/Finger Protocol)
- **Telnet Protocol** (Telnet Protocol)
- **Trivial File Transfer Protocol** (Trivial File Transfer Protocol)

TCP/IP implements other higher–level protocols that are not official Internet protocols but are commonly used in the Internet community at the application program level. These protocols include:

- Distributed Computer Network (DCN) **Local–Network Protocol** (Distributed Computer Network Local–Network Protocol)
- **Remote Command Execution Protocol** (Remote Command Execution Protocol)
- **Remote Login Protocol** (Remote Login Protocol)
- **Remote Shell Protocol** (Remote Shell Protocol)
- **Routing Information Protocol** (Routing Information Protocol)
- **Time Server Protocol** (Time Server Protocol).

TCP/IP does not provide APIs to any of these application–level protocols.

Domain Name Protocol

The **Domain Name Protocol** (**DOMAIN**) allows a host in a domain to act as a *name server* for other hosts within the domain. **DOMAIN** uses **UDP** or **TCP** as its underlying protocol and allows a local network to assign host names within its domain independently from other domains. Normally, the **DOMAIN** protocol uses **UDP**. However, if the **UDP** response is truncated, **TCP** can be used. The **DOMAIN** protocol in **TCP/IP** supports both.

In the **DOMAIN** hierarchical naming system, local resolver routines can resolve Internet names and addresses using a local name resolution database maintained by the **named** daemon. If the name requested by the host is not in the local database, the resolver routine queries a remote **DOMAIN** name server. In either case, if the name resolution information is unavailable, the resolver routines attempt to use the **/etc/hosts** file for name resolution.

Note:: **TCP/IP** configures local resolver routines for the **DOMAIN** protocol if the local file **/etc/resolv.conf** exists. If this file does not exist, the **TCP/IP** configures the local resolver routines to use the **/etc/hosts** database.

TCP/IP implements the **DOMAIN** protocol in the **named** daemon and in the resolver routines and does not provide an API to this protocol.

Exterior Gateway Protocol

Exterior Gateway Protocol (**EGP**) is the mechanism that allows the exterior gateway of an *autonomous system* to share routing information with exterior gateways on other autonomous systems.

Autonomous Systems

An autonomous system is a group of networks and gateways for which one administrative authority has responsibility. Gateways are *interior neighbors* if they reside on the same autonomous system and *exterior neighbors* if they reside on different autonomous systems. Gateways that exchange routing information using **EGP** are said to be **EGP** *peers* or *neighbors*. Autonomous system gateways use **EGP** to provide access information to their **EGP** neighbors.

EGP allows an exterior gateway to ask another exterior gateway to agree to exchange access information, continually checks to ensure that its **EGP** neighbors are responding, and helps **EGP** neighbors to exchange access information by passing routing update messages.

EGP restricts exterior gateways by allowing them to advertise only those destination networks reachable entirely within that gateway's autonomous system. Thus, an exterior gateway using **EGP** passes along information to its **EGP** neighbors but does not advertise access information about its **EGP** neighbors outside its autonomous system.

EGP does not interpret any of the distance metrics that appear in routing update messages from other protocols. **EGP** uses the distance field to specify whether a path exists (a value of 255 means that the network is unreachable). The value cannot be used to compute the shorter of two routes unless those routes are both contained within a single autonomous system. Therefore, **EGP** cannot be used as a routing algorithm. As a result, there will be only one path from the exterior gateway to any network.

In contrast to the **Routing Information Protocol** (**RIP**), which can be used within an autonomous system of Internet networks that dynamically reconfigure routes, **EGP** routes are predetermined in the **/etc/gated.conf** file. **EGP** assumes that **IP** is the underlying protocol.

EGP Message Types

Neighbor Acquisition Request
Used by exterior gateways to request to become neighbors of each other.

Neighbor Acquisition Reply
Used by exterior gateways to accept the request to become neighbors.

Neighbor Acquisition Refusal
Used by exterior gateways to deny the request to become neighbors. The refusal message includes reasons for refusal, such as `out of table space`.

Neighbor Cease
Used by exterior gateways to cease the neighbor relationship. The cease message includes reasons for ceasing, such as `going down`.

Neighbor Cease Acknowledgment
Used by exterior gateways to acknowledge the request to cease the neighbor relationship.

Neighbor Hello
Used by exterior gateways to determine connectivity. A gateway issues a `Hello` message and another gateway issues an `I Heard You` message.

I Heard You
Used by exterior gateways to reply to a `Hello` message. The `I Heard You` message includes the access of the answering gateway and, if the gateway is unreachable, a reason for lack of access, such as `You are unreachable because of problems with my network interface`.

NR Poll
Used by exterior gateways to query neighbor gateways about their ability to reach other gateways.

Network Reachability
Used by exterior gateways to answer the `NR Poll` message. For each gateway in the message, the `Network Reachability` message contains information on the addresses that gateway can reach through its neighbors.

EGP Error
Used by exterior gateways to respond to EGP messages that contain bad checksums or have fields containing incorrect values.

TCP/IP implements the **EGP** protocol in the **gated** server command and does not provide an API to this protocol.

File Transfer Protocol

File Transfer Protocol (**FTP**) allows hosts to transfer data among dissimilar hosts, as well as files between two foreign hosts indirectly. **FTP** provides for such tasks as listing remote directories, changing the current remote directory, creating and removing remote directories, and transferring multiple files in a single request. **FTP** keeps the transport secure by passing user and account passwords to the foreign host. Although **FTP** is designed primarily to be used by applications, it also allows interactive user–oriented sessions.

FTP uses reliable stream delivery (**TCP/IP**) to send the files and uses a Telnet connection to transfer commands and replies. **FTP** also understands several basic file formats including NETASCII, IMAGE, and Local 8.

TCP/IP implements **FTP** in the **ftp** user command and the **ftpd** server command and does not provide an applications programming interface (API) to this protocol.

When creating anonymous ftp users and directories please be sure that the home directory for users ftp and anonymous (for example, **/u/ftp**) is owned by root and does not allow write permissions (for example, `dr–xr–xr–x`). The script **/usr/samples/tcpip/anon.ftp** can be used to create these accounts, files and directories.

Telnet Protocol

The **Telnet Protocol** (**TELNET**) provides a standard method for terminal devices and terminal–oriented processes to interface. **TELNET** is commonly used by terminal emulation programs that allow you to log into a remote host. However, **TELNET** can also be used for terminal–to–terminal communication and interprocess communication. **TELNET** is also used by other protocols (for example, **FTP**) for establishing a protocol control channel.

TCP/IP implements **TELNET** in the **tn, telnet,** or **tn3270** user commands. The **telnetd** daemon does not provide an API to **TELNET**.

TCP/IP supports the following **TELNET** options which are negotiated between the client and server:

BINARY TRANSMISSION (Used in **tn3270** sessions)	Transmits characters as binary data.
SUPPRESS GO_AHEAD (The operating system suppresses GO–AHEAD options.)	Indicates that when in effect on a connection between a sender of data and the receiver of the data, the sender need not transmit a GO_AHEAD option. If the GO_AHEAD option is not desired, the parties in the connection will probably suppress it in both directions. This action must take place in both directions independently.
TIMING MARK (Recognized, but has a negative response)	Makes sure that previously transmitted data has been completely processed.
EXTENDED OPTIONS LIST	Extends the **TELNET** option list for another 256 options. Without this option, the **TELNET** option allows only 256 options.
ECHO (User–changeable command)	Transmits echo data characters already received back to the original sender.
TERM TYPE	Enables the server to determine the type of terminal connected to a user **TELNET** program.

SAK (Secure Attention Key)	Establishes the environment necessary for secure communication between you and the system.
NAWS (Negotiate About Window Size)	Enables client and server to negotiate dynamically for the window size. This is used by applications that support changing the window size.

Note:: **TELNET** must allow transmission of eight bit characters when not in binary mode in order to implement ISO 8859 Latin code page. This is necessary for internationalization of the **TCP/IP** commands.

Trivial File Transfer Protocol

The **Trivial File Transfer Protocol** (**TFTP**) can read and write files to and from a foreign host. Because **TFTP** uses the unreliable User Datagram Protocol to transport files, it is generally quicker than **FTP**. Like **FTP**, **TFTP** can transfer files as either NETASCII characters or as 8–bit binary data. Unlike **FTP**, **TFTP** cannot be used to list or change directories at a foreign host and it has no provisions for security like password protection. Also, data can be written or retrieved only in public directories.

The **TCP/IP** implements **TFTP** in the **tftp** and **utftp** user commands and in the **tftpd** server command. The **utftp** command is a form of the **tftp** command for use in a pipe. **TCP/IP** does not provide an API to this protocol.

Name/Finger Protocol

The **Name/Finger Protocol** (**FINGER**) is an application–level Internet protocol that provides an interface between the **finger** command and the **fingerd** daemon. The **fingerd** daemon returns information about the users currently logged in to a specified remote host. If you execute the finger command specifying a user at a particular host, you will obtain specific information about that user. The **FINGER** Protocol must be present at the remote host and at the requesting host. **FINGER** uses **Transmission Control Protocol** (Transmission Control Protocol) as its underlying protocol.

Note:: **TCP/IP** does not provide an API to this protocol.

Distributed Computer Network Local–Network Protocol

Local–Network Protocol (**HELLO**) is an interior gateway protocol designed for use within autonomous systems. (For more information, see Autonomous Systems on page 4-30.) **HELLO** maintains connectivity, routing, and time–keeping information. It allows each machine in the network to determine the shortest path to a destination based on time delay and then dynamically updates the routing information to that destination.

The **gated** daemon provides the Distributed Computer Network (DCN) local network protocol.

Remote Command Execution Protocol

The **rexec** user command and the **rexecd** daemon provide the remote command execution protocol, allowing users to run commands on a compatible remote host.

Remote Login Protocol

The **rlogin** user command and the **rlogind** daemon provide the **remote login protocol**, allowing users to log in to a remote host and use their terminals as if they were directly connected to the remote host.

Remote Shell Protocol

The **rsh** user command and the **rshd** daemon provide the **remote command shell protocol**, allowing users to open a shell on a compatible foreign host for running commands.

Routing Information Protocol

Routing Information Protocol (**RIP**) and the **routed** and **gated** daemons that implement it keep track of routing information based on gateway hops and maintain kernel–routing table entries.

Time Server Protocol

The **timed** daemon is used to synchronize one host with the time of other hosts. It is based on the client/server concept.

Assigned Numbers

For compatibility with the general network environment, well–known numbers are assigned for the Internet versions, networks, ports, protocols, and protocol options. Additionally, well–known names are also assigned to machines, networks, operating systems, protocols, services, and terminals. **TCP/IP** complies with the assigned numbers and names defined in RFC 1010, *Assigned Numbers*.

The **Internet Protocol** (**IP**)defines a 4–bit field in the **IP** header that identifies the version of the general Internetwork protocol in use. For **IP**, this version number in decimal is 4. For details on the assigned numbers and names used by **TCP/IP**, see **/etc/protocols** and **/etc/services** files included with **TCP/IP**. For further details on the assigned numbers and names, refer to RFC 1010 and the **/etc/services** file.

TCP/IP Local Area Network Adapter Cards

The topics discussed in this section are:

- Installing a Network Adapter on page 4-36
- Configuring and Managing Adapters on page 4-36
- Configuring and Using Virtual Local Area Networks (VLANs) on page 4-38
- Using ATM Adapters on page 4-39

The network adapter card is the hardware that is physically attached to the network cabling. It is responsible for receiving and transmitting data at the physical level. The network adapter card is controlled by the network adapter device driver.

A machine must have one network adapter card (or connection) for each network (not network type) to which it connects. For instance, if a host attaches to two token–ring networks, it must have two network adapter cards.

TCP/IP uses the following network adapter cards and connections:

- Standard Ethernet Version 2
- IEEE 802.3
- Token–ring
- Asynchronous adapters and native serial ports (described in *AIX 5L Version 5.2 Asynchronous Communications Guide*)
- Fiber Distributed Data Interface (FDDI)
- Serial Optical Channel Converter (described in *AIX 5L Version 5.2 Kernel Extensions and Device Support Programming Concepts*)
- Asynchronous Transfer Mode (ATM).

The Ethernet and 802.3 network technologies use the same type of adapter.

Each machine provides a limited number of expansion slots, some or all of which you might wish to use for communications adapters. Additionally, each machine supports a limited number of communications adapters of a given type. Each machine supports up to eight Ethernet/802.3 adapters, up to eight token–ring adapters, and one asynchronous adapter card with up to 64 connections. Within these limits (software limitations), you can install any combination of these adapters up to the total number of expansion slots available in your machine (hardware limitations).

Only one Transmission Control Protocol/Internet Protocol (TCP/IP) interface is configurable regardless of the number of Serial Optical Channel Converters supported by the system. The Serial Optical device driver makes use of both channel converters even though only one logical TCP/IP interface is configured.

Installing a Network Adapter

To install a network adapter:

1. Shut down the computer. See the **shutdown** command for information on how to shut down a system.
2. Turn off the computer power.
3. Remove the computer cover.
4. Find a free slot and insert the network adapter. Be careful to seat the adapter properly in the slot.
5. Replace the computer cover.
6. Restart the computer.

Configuring and Managing Adapters

To configure and manage token–ring or Ethernet adapters, use the tasks in the following table.

Configuring and Managing Adapters Tasks

Task	SMIT fast path	Command or file	Web-based System Manager Management Environment [5]

Configure an Adapter	**smit chgtok** (token ring)
	smit chgenet (Ethernet)
		1. Determine adapter name: [1]
		`lsdev –C –c adapter –t tokenring –H`
		or
		`lsdev –C –c adapter –t ethernet –H`
		2. Reset ring speed (token ring) or connector type (Ethernet), if necessary. For example:
		`chdev –l tok0 –a ring_speed=16 –P`
		or
		`chdev –l ent0 –a bnc_select=dix –P`	

Determining a Network Adapter Hardware Address	**smit chgtok** (token ring) **smit chgenet** (Ethernet)	`lscfg –l tok0 –v` (token ring) @T>2 `lscfg –l ent0 –v` (Ethernet) @T>2	

Setting an Alternate Hardware Address	**smit chgtok** (token ring)
	smit chgenet (Ethernet)
		1. Define the alternate hardware address. For example, for token ring: [2,3]
		`chdev –l tok0 –a alt_addr=0X10005A4F1B7F`
		For Ethernet: [2,3]
		`chdev –l ent0 –a alt_addr=0X10005A4F1B7F –p`
		2. Begin using alternate address, for token ring: [4]
		`chdev –l tok0 –a use_alt_addr=yes`
		For Ethernet: [4]
		`chdev –l ent0 –a use_alt_addr=yes`	

Notes:

1. The name of a network adapter can change if you move it from one slot to another or remove it from the system. If you ever move the adapter, issue the **diag –a** command to update the configuration database.
2. Substitute your adapter name for `tok0` and `ent0`.
3. Substitute your hardware address for `0X10005A4F1B7F`.
4. After performing this procedure, you might experience a disruption of communication with other hosts until they flush their Address Resolution Protocol (ARP) cache and obtain the new hardware address of this host.
5. These tasks are not available in Web-based System Manager Management Environment.

Configuring and Using Virtual Local Area Networks (VLANs)

VLANs (Virtual Local Area Networks) can be thought of as logical broadcast domains. A VLAN splits up groups of network users on a real physical network onto segments of logical networks. This implementation supports the IEEE 802.1Q VLAN tagging standard with the capability to support multiple VLAN IDs running on Ethernet adapters. Each VLAN ID is associated with a separate Ethernet interface to the upper layers (IP, etc.) and creates unique logical Ethernet adapter instances per VLAN, for example `ent1`, `ent2` and so on.

The IEEE 802.1Q VLAN support can be configured over any supported Ethernet adapters. The adapters must be connected to a switch that supports IEEE 802.1Q VLAN.

You can configure multiple VLAN logical devices on a single system. Each VLAN logical devices constitutes an additional Ethernet adapter instance. These logical devices can be used to configure the same Ethernet IP interfaces as are used with physical Ethernet adapters. As such, the **no** option, *ifsize* (default 8), needs to be increased to include not only the Ethernet interfaces for each adapter, but also any VLAN logical devices that are configured. See the **no** command documentation.

Each VLAN can have a different maximum transmission unit (MTU) value even if sharing a single physical Ethernet adapter.

VLAN support is managed through SMIT. Type the **smit vlan** fast path from the command line and make your selection from the main VLAN menu. Online help is available.

After you configure VLAN, configure the IP interface, for example, `en1` for standard Ethernet or `et1` for IEEE 802.3, using Web–based System Manager, SMIT, or commands. Notes:

1. If you try to configure a VLAN ID value that is already in use for the specified adapter, the configuration fails with the following error:

   ```
   Method error (/usr/lib/methods/chgvlan):
      0514–018 The values specified for the following attributes
               are not valid:
                vlan_tag_id          VLAN Tag ID
   ```

2. If a user (for example, IP interface) is currently using the VLAN logical device, any attempt to remove the VLAN logical device fails. A message similar to the following displays:

   ```
   Method error (/usr/lib/methods/ucfgcommo):
            0514–062 Cannot perform the requested function because the
                     specified device is busy.
   ```

   To remove the logical VLAN device, first detach the user. For example, if the user is IP interface `en1`, then you can use the following command:

   ```
   ifconfig en1 detach
   ```

   Then remove the network interface using the SMIT TCP/IP menus.

3. If a user (for example, IP interface) is currently using the VLAN logical device, any attempt to change the VLAN characteristic (VLAN tag ID or base adapter) fails. A message similar to the following displays:

   ```
   Method error (/usr/lib/methods/chgvlan):
            0514–062 Cannot perform the requested function because the
                     specified device is busy.
   ```

   To change the logical VLAN device, first detach the user. For example, if the user is the IP interface `en1`, you could use the following command:

   ```
   ifconfig en1 detach
   ```

   Then change the VLAN and add the network interface again using the SMIT TCP/IP menus.

Troubleshooting

tcpdump and **trace** can be used to troubleshoot the VLAN. The trace hook ID for each type of transmit packet follows:

transmit packets	3FD
receive packets	3FE
other events	3FF

The **entstat** command gives the aggregate statistics of the physical adapter for which the VLAN is configured. It does *not* provide the individual statistics for that particular VLAN logical device.

Restrictions

Remote dump is not supported over a VLAN. Also, VLAN logical devices cannot be used to create a Cisco Systems’ Etherchannel.

Using ATM Adapters

Asynchronous Transfer Mode (ATM) is an international standard that defines a high–speed networking method to transport any mixture of voice, video, and traditional computer data across local, municipal, and wide–area networks (LANs, MANs, and WANs). ATM adapters provide full–duplex connectivity for ESCALA servers or clients using permanent virtual circuits (PVCs) and switched virtual circuits (SVCs). The PVC and SVC implementations are designed to be compliant with the ATM Forum specifications. The maximum number of virtual circuits supported depends on the adapter. Most adapters support at least 1024 virtual circuits.

ATM Technology

Asynchronous Transfer Mode (ATM) is a cell–switching, connection–oriented technology. In ATM networks, end stations attach to the network using dedicated full duplex connections. The ATM networks are constructed using switches, and switches are interconnected using dedicated physical connections. Before any data transfers can begin, end–to–end connections must be established. Multiple connections can and do exist on a single physical interface. Sending stations transmit data by segmenting Protocol Data Units (PDUs) into 53–byte cells. Payload stays in the form of cells during network transport. Receiving stations reassemble cells into PDUs. The connections are identified using a virtual path identifier (VPI) and a virtual channel identifier (VCI). The VPI field occupies one byte in the ATM cell five–byte header; whereas, the VCI field occupies two bytes in the ATM cell five–byte header. Basically, a VPI:VCI pair identifies the source of the ATM cell. The function of the ATM switch is to recognize the source of the cell, determine the next hop, and output the cell to a port. The VPI:VCI changes on a hop–by–hop basis. Thus, VPI:VCI values are not universal. Each virtual circuit is described as a concatenation of VPI:VCI values across the network.

ATM Connections

ATM architecture has two kinds of virtual circuits: permanent (PVCs) and switched (SVCs).

Permanent Virtual Circuits
PVCs are statically and manually configured. The switches forming the ATM network must first be set up to recognize the VPI:VCI combination of each endpoint and to route the endpoint ATM cells to the destination endpoint through the ATM network. Once a link connection through the network has been established from one endpoint to another, ATM cells can be transmitted through the ATM network and ATM switches. The network switches translate the VPI:VCI values in the appropriate way so as to route the cell to its destination.

Switched Virtual Circuits
SVCs are dynamically set up on an as needed basis. The ATM end stations are assigned 20–byte addresses. SVCs use a control plane and a data plane.
The control plane uses a signaling channel VPI:VCI 0:5.
SVCs involve on demand call setup, whereby an ATM station sends information elements specifying the destination ATM address (and optionally, the source ATM address). In general, calling station, network, and called station participate in a negotiation. Finally, a call is either accepted or rejected. If a call is accepted, network assigns VPI:VCI values for the data plane to the calling station and called station. In the control plane, the ATM network routes (or switches) signaling packets on the basis of the ATM addresses. While these packets are being routed, the switches set up data plane cell routing tables. In the data plane, ATM networks switch cells on the basis of VPI:VCI much like in the case of PVCs. When data transfer is over, connection is terminated.

The ATM address is constructed by registering with the ATM network and by acquiring the most significant 13 bytes. The next six bytes contain the adapter’s factory–assigned, unique MAC address. The least significant byte is the selector. Use of this byte is left to the discretion of the end station. ATM networks do not interpret this byte.

TCP/IP over ATM

The *Internet Engineering Task Force RFC1577: Classical IP and ARP over ATM* standard specifies the mechanism for implementing Internet Protocol (IP) over ATM. Since ATM is connection–oriented technology and IP is a datagram–oriented technology, mapping the IP over ATM is not trivial.

In general, the ATM network is divided into logical IP subnetworks (LISs). Each LIS is comprised of some number of ATM stations. LISs are analogous to traditional LAN segments. LISs are interconnected using routers. A particular adapter (on an ATM station) can be part of multiple LISs. This feature can be very useful for implementing routers.

RFC1577 specifies RFC1483, which specifies logical link control/Sub–Network Access Protocol (LLC/SNAP) encapsulation as the default. In PVC networks for each IP station, all PVCs must be manually defined by configuring VPI:VCI values. If LLC/SNAP encapsulation is not being used, the destination IP address associated with each VPI:VCI must be defined. If LLC/SNAP encapsulation is being used, the IP station can learn the remote IP address by an InARP mechanism.

For SVC networks, RFC1577 specifies an ARP server per LIS. The purpose of the ARP server is to resolve IP addresses into ATM addresses without using broadcasts. Each IP station is configured with the ATM address of the ARP server. IP stations set up SVCs with the ARP server, which in turn, sends InARP requests to the IP stations. Based on InARP reply, an ARP server sets up IP to ATM address maps. IP stations send ARP packets to the ARP server to resolve addresses, which returns ATM addresses. IP stations then set up a SVC to the destination station and data transfer begins. The ARP entries in IP stations and the ARP server age based on a well defined mechanism. For both the PVC and SVC environments, each IP station has at least one virtual circuit per destination address.

The Internet Engineering Task Force RFC2225 adds the support of ATM ARP Request Address list to RFC1577. The ATM ARP Request Address list is a list containing one or more ATM addresses of individual ATM ARP servers located within the LIS. The RFC2225 client eliminates the single point of failure associated with the 1577 clients' ATM ARP services. The 2225 clients have the ability to switch to backup ARP servers when the current ATM ARP server fails.

ESCALA sets the first entry in the ATM ARP Request Address list as the Primary ATM ARP server and the rest of the entries as Secondary ATM ARP servers.

The client will always try to use the Primary ATM ARP server. If the effort to connect to the Primary ATM ARP server fails, the client tries to connect to the first Secondary server (the position in the ATM ARP Request Address list determines the order of the Secondary ATM ARP server). If the connection to the first Secondary ATM ARP server fails, the client tries to contact the next Secondary ATM ARP server in the list. This process continues until the connection is successful.

If the connection to the Primary ATM ARP server fails, regardless of which Secondary ATM ARP server it is connected to or attempting to connect to, the client continues to retry the Primary ATM ARP server every 15 minutes. If it finally connects to the Primary ATM ARP server, then the connection to the current Secondary ATM ARP server is dropped.

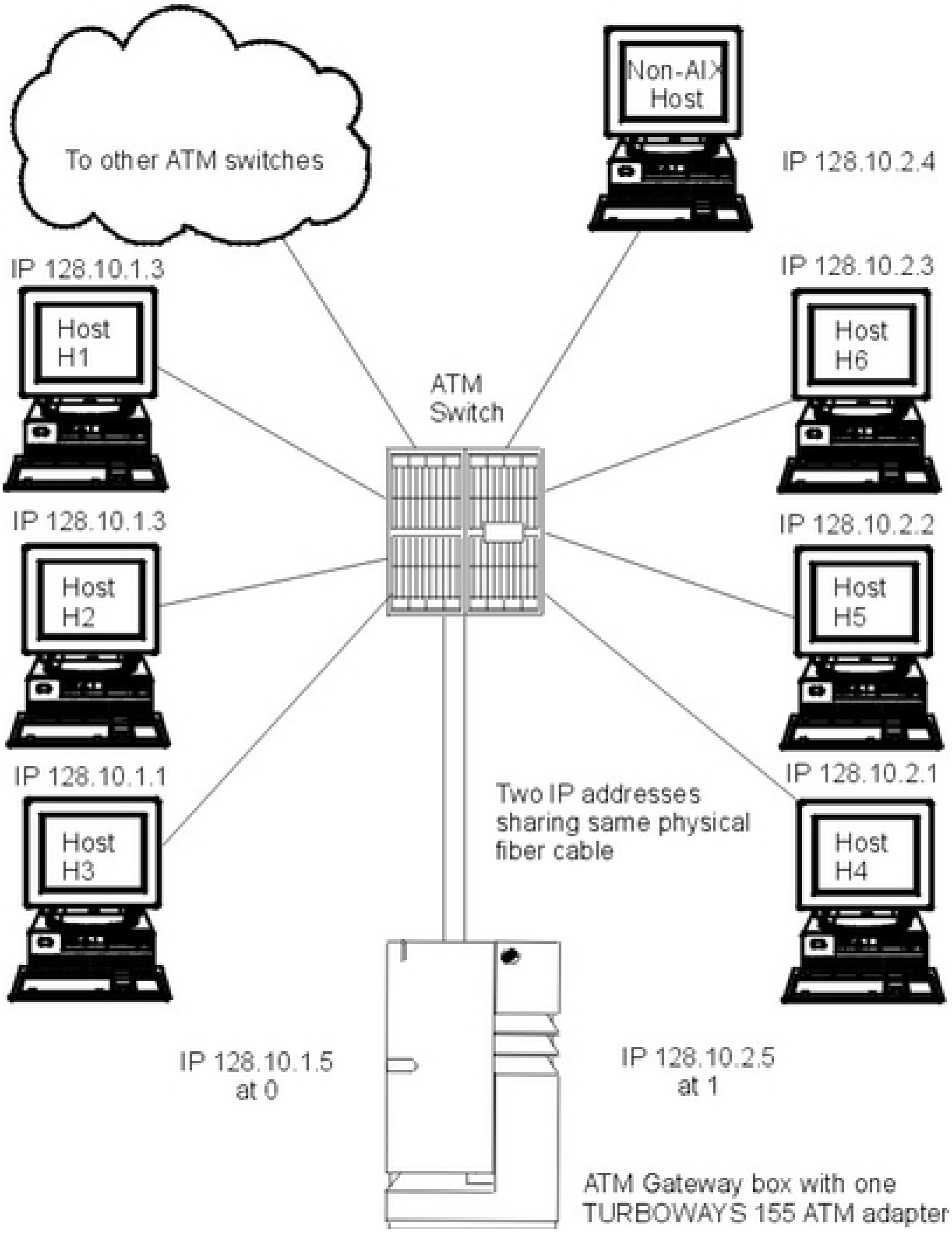
The ATM ARP Request Address list is entered manually either through SMIT or by using the **ifconfig** command. The ATM ARP Request Address list cannot be configured with the Management Information Base (MIB).

PVC Network

Use Figure 14 as an example to configure your network.

Within the ”Representative ATM Network” figure, one logical IP subnet is represented by dashed lines from each host to the switch. The other IP subnet is represented by solid lines from each host to the switch.

Figure 14. Representative ATM Network This illustration depicts an ATM network laid out in a typical star topography. In the center of the star is the ATM switch. Numbered IP hosts are branched off of the switch as are links to other ATM switches and one ATM gateway box and adapter.



The following Representative Host Configuration table indicates how hosts H3 and H4 are configured to communicate with a gateway and with each host on its own logical IP subnet.

Representative Host Configuration

Network Interface Driver	VPI:VCI	Comment
Host H3		
at0	0:40	Connection to 128.10.1.5 (gateway)
at0	0:42	Connection to 128.10.1.2
at0	0:43	Connection to 128.10.1.3
Host H4		
at0	0:50	Connection to 128.10.2.5 (gateway)
at0	0:52	Connection to 128.10.2.2
at0	0:53	Connection to 128.10.2.3
at0	0:54	Connection to 128.10.2.4

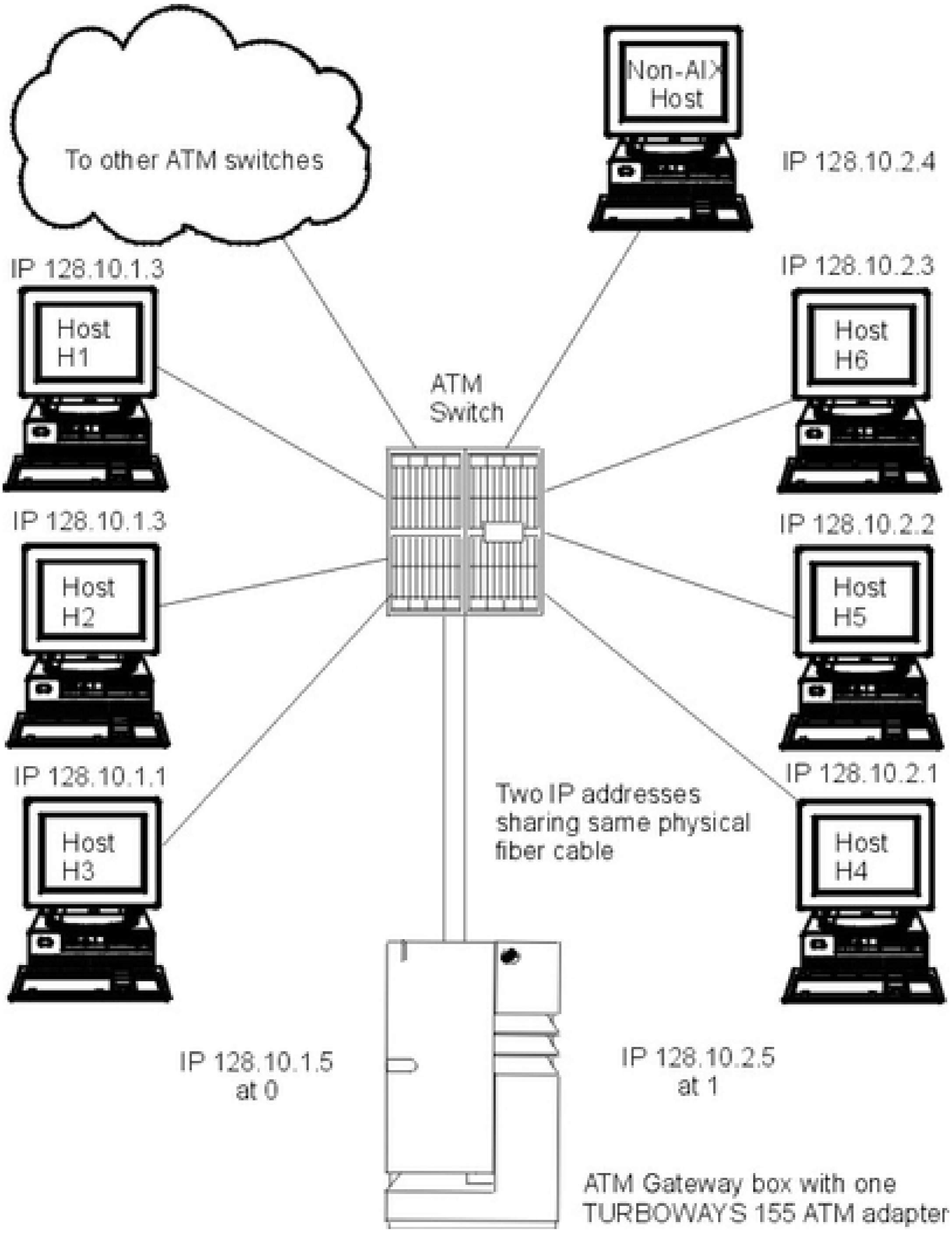
To reach hosts on another logical IP subnet, only a VPI:VCI connection to the gateway needs to be created. (The VPI:VCIs are for illustration purposes only.)

The ATM gateway box has one ATM with two IP addresses sharing the same physical cable.

SVC Network

Using Figure 15 as an example, imagine that host H3 wants to call H4. H1 is the ARP server for subnet 1 and H6 is the ARP server for subnet 2. Assuming a subnet mask of 255.255.255.0, stations with addresses of 128.10.1.X are members of one subnet, whereas stations with addresses of 128.10.2.X are members of a second subnet. See the following list of representative host configurations using SVCs.

Figure 15. Representative ATM Network This illustration depicts an ATM network laid out in a typical star topography. In the center of the star is the ATM switch. Numbered IP hosts are branched off of the switch as are links to other ATM switches and one ATM gateway box and adapter.



List of Representative Host Configurations

Network interface driver	IP address	ARP server	ARP server address	Gateway address
Host H1				
at0	128.10.1.3	Yes		128.10.1.5
Host H3				
at0	128.10.1.1	No	ATM address of H1	128.10.1.5
Gateway				
at0	128.10.1.5	No	ATM address of H1	
at1	128.10.2.5	No	ATM address of H6	
Host H4				
at0	128.10.2.1	No	ATM address of H6	128.10.2.5
Host H6				
at0	128.10.2.3	Yes		128.10.2.5

Note:: Each subnet requires one and only one ARP server.

Because H3 recognizes that address 128.10.2.1 is not on its subnet, H3 consults H1 to resolve the default gateway IP address to an ATM address. H3 then places a call to the gateway. The gateway recognizes that the data is bound for the second subnet and consults H6 to successfully resolve the H4 IP address to an ATM address. Connections are then established between H3 and the gateway and between the gateway and H4.

Configuring an ATM Adapter

To configure your ATM adapter, use the Web–based System Manager, **wsm**, or the SMIT fast path **smit chg_atm**. Select an adapter name, then use the online help and multiple–choice lists to decide which changes to make for your configuration.

ATM Adapter Statistics

The **atmstat** command can be used for getting ATM adapter statistics. Using the **atmstat** command with the **–r** flag resets the statistics. The format of the command is **atmstat** *DeviceName*. This command returns the following sets of statistics:

Transmit Statistics

`Packets:` This field contains the number of packets (or PDUs) transmitted.

`Bytes:` This field contains the count of bytes transmitted. These are the user bytes. The ATM overhead, for example, ATM cell header, and AAL 5 PDU trailer, are excluded.

`Interrupts:` This field is not used.

`Transmit Errors:`
This field contains the number of transmit errors for this device.

`Packets Dropped:`
This field contains the number of Transmit Packets that were dropped, for instance, because of an out of buffers condition.

`Max Packets on S/W Transmit Queue:`
This field does not apply to ATM.

`S/W Transmit Queue Overflow:`
This field does not apply to ATM.

`Current S/W + H/W Transmit Queue Length:`
This field contains the current transmit queue length.

`Cells Transmitted:`
This field contains the number of cells transmitted by this device.

`Out of Xmit Buffers:`
This field contains the number of packets dropped because of out of xmit buffers condition.

`Current HW Transmit Queue Length:`
This field contains the current number of transmit packets on the hardware queue.

`Current SW Transmit Queue Length:`
This field does not apply to ATM.

Receive Statistics

`Packets:` This field contains the number of packets (or PDUs) received.

`Bytes:` This field contains the count of bytes received. These are the user bytes. The ATM overhead, for example, ATM cell header and AAL 5 PDU trailer are excluded.

`Interrupts:` This field contains the number of Interrupts taken by the system for the adapter–to–system indications. Some of the events that cause these interrupts are packet received, transmit done indication, and so on.

`Receive Errors:`
This field contains the number of receive errors for this device.

`Packets Dropped:`
This field contains the number of received packets dropped, for instance, due to out of buffers condition.

`Bad Packets:`
This field does not apply to ATM.

`Cells Received:`
This field contains the number of cells received by this device.

`Out of Rcv Buffers:`
This field contains the number of packets dropped because of out of receive buffers condition.

`CRC Errors:` This field contains the number of received packets that encountered CRC errors.

`Packets Too Long:`
This field contains the number of received packets that exceeded the maximum PDU size.

`Incomplete Packets:`
This field contains the number of incomplete received packets.

`Cells Dropped:`
This field contains the number of dropped cells. Cells could be dropped for a number of reasons, such as bad header error control (HEC), out of buffer condition, and so on.

General Statistics

`No mbuf Errors:`

This field contains the number of mbuf requests that were denied.

`Adapter Loss of Signals:`

This field contains the number of times the adapter encountered loss of signal.

`Adapter Reset Count:`

This field contains the number of times the adapter has been reset.

`Driver Flags: Up Running Simplex`

This field contains the neighborhood discovery daemon (NDD) flags.

`Virtual Connections in use:`

This field contains the number of virtual connections that are currently allocated or in use.

`Max Virtual Connections in use:`

This field contains the maximum number of virtual connections allocated since the last reset of the statistics.

`Virtual Connections Overflow:`

This field contains the number of allocate virtual connections requests that have been denied.

`SVC UNI Version:`

This field contains the current UNI version of the signaling protocol in use.

Additional Microchannel ATM Statistics

Note:: The following Microchannel information applies to AIX 5.1 and earlier.

Using the **atmstat** command with the **–d** flag provides detailed statistics.

`Packets Dropped – No small direct memory access (DMA) buffer:`

This field contains the number of received packets dropped because the adapter did not have small system buffers for DMA.

`Packets Dropped – No medium DMA buffer:`

This field contains the number of received packets dropped because the adapter did not have medium system buffers for DMA.

`Packets Dropped – No large DMA buffer:`

This field contains the number of received packets dropped because the adapter did not have large system buffers for DMA.

`Receive Aborted – No Adapter Receive buffer:`

This field contains the number of received packets aborted because the adapter did not have on–card receive buffers.

`Transmit Aborted – No small DMA buffer:`

This field contains the number of transmit packets dropped because of the lack of small system buffers for DMA.

`Transmit Aborted – No medium DMA buffer:`

This field contains the number of transmit packets dropped because of the lack of medium system buffers for DMA.

`Transmit Aborted – No large DMA buffer:`

This field contains the number of transmit packets dropped because of the lack of large system buffers for DMA.

`Transmit Aborted – No MTB DMA buffer:`

This field contains the number of transmit packets dropped because of the lack of large system buffers for DMA.

`Transmit Aborted – No Adapter Transmit buffer:`

This field contains the number of transmit packets dropped because of the lack of adapter on–card transmit buffers.

`Max Hardware Transmit Queue Length:`

This field contains the maximum number of transmit packets queued in the hardware queue.

`Small Mbufs in Use:`

This field contains the number of small mbufs currently in use. The adapter device driver allocates these buffers according to the configuration information provided by system administrators. This information can be used to tune the configuration information.

`Medium Mbufs in Use:`

This field contains the number of medium mbufs currently in use. The adapter device driver allocates these buffers according to the configuration information provided by system administrators. This information can be used to tune the configuration information.

`Large Mbufs in Use:`

This field contains the number of large mbufs currently in use. The adapter device driver allocates these buffers according to the configuration information provided by system administrators. This information can be used to tune the configuration information.

`Huge Mbufs in Use:`

This field contains the number of huge mbufs currently in use. The adapter device driver allocates these buffers according to the configuration information provided by the system administrators. This information can be used to tune the configuration information.

`MTB Mbufs in Use:`

This field contains the number of MTB mbufs currently in use. The adapter device driver allocates these buffers according to the configuration information provided by the system administrators. This information can be used to tune the configuration information.

`Max Small Mbufs in Use:`

This field contains the maximum number of small mbufs that have been used. The adapter device driver allocates these buffers according to the configuration information provided by the system administrators. This information can be used to tune the configuration information.

`Max Medium Mbufs in Use:`

This field contains the maximum number of medium mbufs that have been used. The adapter device driver allocates these buffers according to the configuration information provided by system administrators. This information can be used to tune the configuration information.

`Max Large Mbufs in Use:`

This field contains the maximum number of large mbufs that have been used. The adapter device driver allocates these buffers according to the configuration information provided by system administrators. This information can be used to tune the configuration information.

`Max Huge Mbufs in Use:`

This field contains the maximum number of huge mbufs that have been used. The adapter device driver allocates these buffers according to the configuration information provided by system administrators. This information can be used to tune the configuration information.

`MTB Mbufs in Use:`
This field contains the maximum number of MTB mbufs that have been used. The adapter device driver allocates these buffers according to the configuration information provided by system administrators. This information can be used to tune the configuration information.

`Small Mbufs overflow:`
This field contains the number of times that a small mbuf could not be allocated. This information can be used to tune the configuration information.

`Medium Mbufs overflow:`
This field contains the number of times that a medium mbuf could not be allocated. This information can be used to tune the configuration information.

`Large Mbufs overflow:`
This field contains the number of times that a large mbuf could not be allocated. This information can be used to tune the configuration information.

`Huge Mbufs overflow:`
This field contains the number of times that a huge mbuf could not be allocated. This information can be used to tune the configuration information.

`MTB Mbufs overflow:`
This field contains the number of times that an MTB mbuf could not be allocated. This information can be used to tune the configuration information.

PCI ATM Adapter Specific Statistics

`Total 4K byte Receive Buffers: 768 Using: 512`
This message contains the number of receive buffers allocated as well as the number that are currently in use.

`Max 4K byte Receive Buffers limit: 1228 max_used: 514`
This message contains the maximum number of receive buffers that can be allocated as well as the number that have been used since the adapter was last configured or opened.

TCP/IP Network Interfaces

The TCP/IP Network Interface layer formats IP datagrams at the Network layer into packets that specific network technologies can understand and transmit. A network interface is the network–specific software that communicates with the network–specific device driver and the IP layer in order to provide the IP layer with a consistent interface to all network adapters that might be present.

The IP layer selects the appropriate network interface based on the destination address of the packet to be transmitted. Each network interface has a network address. The Network Interface layer is responsible for adding or removing any link layer protocol header required to deliver a message to its destination. The **network adapter** device driver controls the network adapter card.

Although not required, a network interface is usually associated with a network adapter. For instance, the loopback interface has no network adapter associated with it. A machine must have one network adapter card for each network (not network type) to which it connects. However, a machine requires only one copy of the network interface software for each network adapter it uses. For instance, if a host attaches to two token–ring networks, it must have two network adapter cards. However, only one copy of the **token–ring** network interface software and one copy of the token–ring device driver is required.

TCP/IP supports types of network interfaces:

- Standard Ethernet Version 2 (en)
- IEEE 802.3 (et)
- Token–ring (tr)
- Serial Line Internet Protocol (SLIP)
- Loopback (lo)
- FDDI
- Serial Optical (so)
- ATM (at)
- Point–to–Point Protocol (PPP)
- Virtual IP Address (vi)

The Ethernet, 802.3, and token–ring interfaces are for use with local area networks (LANs). The SLIP interface is for use with serial connections. The loopback interface is used by a host to send messages back to itself. The Serial Optical interface is for use with optical point–to–point networks using the Serial Optical Link device handler. The ATM interface is for use with 100 Mbits/sec and 155 Mbits/sec ATM connections. Point to Point protocol is most often used when connecting to another computer or network via a modem. The Virtual IP Address interface (also called *virtual interface*) is not associated with any particular network adapter. Multiple instances of a virtual interface can be configured on a host. When virtual interfaces are configured, the address of the first virtual interface becomes the source address unless an application has chosen a different interface. Processes that use a virtual IP address as their source address can send packets through any network interface that provides the best route for that destination. Incoming packets destined for a virtual IP address are delivered to the process regardless of the interface through which they arrive.

Automatic Configuration of Network Interfaces

When a new network adapter is physically installed in the system, the operating system automatically adds the appropriate network interface for that adapter. For example, if you install a token–ring adapter in your system, the operating system assigns it the name `tok0` and add a token–ring network interface named `tr0`. If you install an Ethernet adapter in your system, the operating system assigns it the name `ent0` and add both an Ethernet Version 2 and an IEEE 802.3 interface, named `en0` and `et0` respectively.

In most cases, there is a one–to–one correspondence between adapter names and network interface names. For example, token–ring adapter `tok0` corresponds to interface `tr0`, adapter tok1 corresponds to interface `tr1`, and so on. Similarly, Ethernet adapter `ent0` corresponds to interface `en0` (for Ethernet Version 2) and `et0` (for IEEE 802.3), and adapter `ent1` corresponds to interface `en1` (for Ethernet Version 2) and `et1` (for IEEE 802.3).

In the case of ATM, according to RFC1577, it is possible for an ATM station to be part of multiple Logical IP Subnetworks. In this case, multiple interfaces are associated with a device. This requires that an interface be specifically added and a device name be assigned to it.

Note:: Under normal circumstances, you do not need to delete or add a network interface manually. However, some problem determination procedures might require you to do so. In this case, use the Web-based System Manager **wsm**, or the SMIT fast path, **smit inet**, to delete and re–add the appropriate interface.

At each system startup, the operating system automatically configures the network interface software based upon the information in the ODM database. Initially, the network interface is configured with default values. In order to communicate through a given network interface, the Internet address must be set. This is the only attribute that you need to set. All other necessary attributes can use the default values. The default values for each network interface follow.

Ethernet Default Configuration Values

The following is a list of valid Ethernet network adapter attributes along with their default values, which can be changed using the Web–based System Manager, **wsm** or the Network Interface Selection menu in SMIT.

Attribute	Default value	Possible values
netaddr		
state	down	up, down, detach
arp	yes	yes, no
netmask		
broadcast		

The following valid Ethernet network device driver attribute is shown along with its default values, which can be changed using the Web–based System Manager, **wsm**, or the Network Interface Drivers menu in SMIT.

Attribute	Default value	Possible values
mtu	1500	60 through 1500

802.3 Default Configuration Values

The following is a list of valid 802.3 network adapter attributes along with their default values, which can be changed using the Web–based System Manager, **wsm**, or the Network Interface Selection menu in SMIT.

Attribute	**Default value**	**Possible values**
netaddr		
state	down	up, down, detach
arp	yes	yes, no
netmask		
broadcast		

The following valid 802.3 network device driver attribute is shown along with its default values, which can be changed using the Web–based System Manager, **wsm**, or the Network Interface Drivers menu in SMIT.

Attribute	**Default value**	**Possible values**
mtu	1492	60 through 1492

Token–Ring Default Configuration Values

The following is a list of valid token–ring network adapter attributes along with their default values, which can be changed using the Web–based System Manager, **wsm**, or the Network Interface Selection menu in SMIT.

Attribute	**Default value**	**Possible values**
netaddr		
netmask		
state	down	up, down, detach
arp	yes	yes, no
hwloop	no	yes, no
netmask		
broadcast		
allcast	no	yes, no

The following valid token–ring network device driver attributes are shown along with its default values, which may be changed using the Web–based System Manager, **wsm**, or the Network Interface Drivers menu in SMIT.

Attribute	**Default value**	**Possible values**
mtu (4Mbps)	1500	60 through 4056
mtu (16Mbps)	1500	60 through 17960

Note:: When operating through a bridge, the default value of 1500 for the maximum transmission unit (MTU) should be changed to a value that is 8 less than the maximum information field (maximum I–frame) advertised by the bridge in the routing control field. For example, if the maximum I–frame value is 1500 in the routing control field, the MTU size should be set to 1492. This is for token–ring network interfaces only. For more information, see Problems with a Token–Ring/Token–Ring Bridge on page 4-239.

When using the IBM16/4PowerPC token–ring adapter (ISA), the MTU is restricted to 2000.

SLIP Default Configuration Values

The following is a list of valid SLIP network adapter attributes along with their default values as shown under the Web–based System Manager, **wsm**, or the Network Interface Selection menu in SMIT.

Attribute	Default value	Possible values
netaddr		
dest		
state	up	up, down, detach
netmask		

The following valid SLIP network device driver attribute is shown along with its default values as displayed under the Web–based System Manager, **wsm**, or the Network Interface Drivers menu in SMIT.

Attribute	Default value	Possible values
mtu	1006	60 through 4096

Serial Optical Default Configuration Values

The following is a list of valid Serial Optical network channel converter attributes along with their default values as shown under the Web–based System Manager, **wsm**, or the Network Interface Selection menu in SMIT.

Attribute	Default value	Possible values
netaddr		
state	down	up, down, detach
netmask		

The following valid serial optical network device handler attribute is shown along with its default values as displayed under the Web–based System Manager, **wsm**, or the Network Interface Drivers menu in SMIT.

Attribute	Default value	Possible values
mtu	61428	1 through 61428

ATM Default Configuration Values

The following is a list of valid ATM network adapter attributes along with their default values as shown under the Web–based System Manager, **wsm**, or the Network Interface Selection menu in SMIT.

Attribute	Default value	Possible values
netaddr		
netmask		
state	up	up, down, detach
Connection Type	svc_s	svc_c, svc_s, pvc
ATM Server Address		
Alternate Device		
idle timer	60	1 through 60
Best Effort Bit Rate (UBR) in kbits/sec	0	1 through 155,000

The following valid ATM network device driver attribute is shown along with its default values as displayed under the Web–based System Manager, **wsm**, or the Network Interface Drivers menu in SMIT.

Attribute	**Default value**	**Possible values**
mtu	9180	1 through 64K

Note:: Network Administrators need to exercise caution while changing MTU size from default to some other value. The MTU size needs to be coordinated with other stations on the network.

If PVCs are to be used on an interface, the VPI:VCIs needs to be defined. This is performed through the Network Interface Selection Menu. The `PVCs for IP over ATM Network` option on this menu is used to list, add, change, or remove PVCs.

Implications of Multiple Network Interfaces on the Same Network

If multiple network interfaces are attached to a single network, each interface must have a unique IP address.

Prior to AIX 5.1, if you configured multiple network interfaces on the same network, only the first interface you configured would have a route to the network in the IP routing table. This means that all outgoing IP traffic would go through that interface only, not any of the other interfaces on that network. Although it is possible to use this configuration to load–balance incoming traffic, it is not a recommended configuration prior to AIX 5.1.

In AIX 5.1 and above, the Multipath Routing feature allows routes to be added to the IP routing table for multipath interfaces on the same subnet. This allows outgoing traffic to alternate between the interfaces rather than being sent through one interface only.

Managing Network Interfaces

To manage network interfaces, use the web–based system manager, WSM Network, FastPath (application) or the tasks in the following table.

Managing network interfaces tasks

Task	**SMIT fast path**	**Command or file**	Web-based System Manager Management Environment
List all network devices	**smit lsinet**	**lsdev –C –c if**	Software —> **Devices** —> **All Devices**.
Configure a network device	**smit chinet**	See the **ifconfig** command and the **rc.net** file	Software —> **Network** —> **TCPIP (IPv4 and IPv6)** —> **Protocol Configuration** —> **Set up basic TCP/IP configuration**.

Changing network interface info with remotely mounted **/usr**	**smit chdev** 1,2	**chgif** 1,2	Software —> **Network** —> **TCPIP (IPv4 and IPv6)** —> **Network Interfaces** —>. Right–click and select **Properties** —> **Aliases**.
Obtaining statistics for a network interface		**netstat –v**	Software —> **Network** —> **TCPIP (IPv4 and IPv6)** —> **Network Interfaces** —> **Network Statistics**.

Notes:

1. Changes from a remotely mounted **/usr** affect only the Information Database (ODM) until the network is restarted or until the **ifconfig** command is used to make the changes take effect right away.
2. When using a remotely mounted **/usr**, be careful not to modify the interface being used, because that is the location of the libraries, commands, and kernel.

Interface–Specific Network Options

TCP/IP interfaces must be specially tuned to achieve good, high–speed network performance (100 Mb or more). This effort is complicated by the fact that multiple network interfaces and a combination of traditional and high–speed TCP/IP interfaces can be used on a single system. Before AIX 4.3.3 (4330–08) and AIX 5.1, AIX provided a single set of system–wide values for the key IP interface network tuning parameters making it impossible to tune a system that has widely differing network adapter interfaces. Beginning with AIX 4.3.3 (4330–08) and AIX 5.1, Interface Specific Network Options (ISNO) allows system administrators to tune each TCP/IP interface individually for best performance.

There are five ISNO parameters for each supported interface: **rfc1323**, **tcp_nodelay**, **tcp_sendspace**, **tcp_recvspace**, and **tcp_mssdflt**. When set, the values for these parameters override the system–wide parameters of the same names that had been set with the **no** command. When ISNO options are not set for a particular interface, system–wide options are used. When options have been set by an application for a particular socket using the **setsockopt** subroutine, such options override the ISNOs.

The network option **use_isno**, set with the **no** command, must have a value of 1 for the ISNOs to take effect. The default value for **use_isno** is 1.

Some high–speed adapters have ISNO parameters set by default in the ODM database.

Gigabit Ethernet interfaces, when configured to use an MTU of 9000, use the following ISNO values by default:

Name	AIX 4.3.3 Value	AIX 4.3.3 (4330–08) Value	AIX 5.1 (and later) Value
tcp_sendspace	131072	262144	262144
tcp_recvspace	92160	131072	131072
rfc1323	1	1	1

Gigabit Ethernet interfaces, when configured to use an MTU of 1500, use the following ISNO values by default:

Name	AIX 4.3.3 Value	AIX 4.3.3 (4330–08) Value	AIX 5.1 (and later) Value
tcp_sendspace	65536	131072	131072
tcp_recvspace	16384	65536	65536
rfc1323	0	not set	not set

ATM interfaces, when configured to use an MTU of 1500, use the following ISNO values by default:

Name	AIX 4.3.3 Value	AIX 4.3.3 (4330–08) Value	AIX 5.1 (and later) Value
tcp_sendspace	16384	not set	not set
tcp_recvspace	16384	not set	not set
rfc1323	0	not set	not set
tcp_nodelay	0	not set	not set
tcp_mssdflt	512	not set	not set

ATM interfaces, when configured to use an MTU of 65527, use the following ISNO values by default:

Name	AIX 4.3.3 Value	AIX 4.3.3 (4330–08) Value	AIX 5.1 (and later) Value
tcp_sendspace	655360	655360	655360
tcp_recvspace	655360	655360	655360
rfc1323	0	1	1
tcp_nodelay	0	not set	not set
tcp_mssdflt	512	not set	not set

ATM interfaces, when configured to use an MTU of 9180, use the following ISNO values by default:

Name	AIX 4.3.3 Value	AIX 4.3.3 (4330–08) Value	AIX 5.1 (and later) Value
tcp_sendspace	65536	65536	65536
tcp_recvspace	65536	65536	65536
rfc1323	0	not set	not set
tcp_nodelay	0	not set	not set
tcp_mssdflt	512	not set	not set

FDDI interfaces, when configured to use an MTU of 4352, use the following ISNO values by default:

Name	Value
tcp_sendspace	45046
tcp_recvspace	45046

The ISNO parameters cannot be displayed or changed using SMIT. They can be set using the **chdev** command or the **ifconfig** command. The **ifconfig** command changes the values only until the next reboot. The **chdev** command changes the values in the ODM database so they are used on subsequent reboots. The **lsattr** or **ifconfig** commands can be used to display the current values.

Example

The following commands can be used first to verify system and interface support and then to set and verify the new values.

1. Verify general system and interface support using the **no** and **lsattr** commands.

   – Ensure the **use_isno** option is enabled using a command similar to the the following:

   ```
   $   no –a | grep isno
              use_isno=1
   ```

   – Ensure the interface supports the five new ISNOs using the **lsattr –El** command, as shown in the following:

   ```
   $   lsattr –E –l en0 –H
              attribute   value  description
              rfc1323                     N/A
              tcp_nodelay                 N/A
              tcp_sendspace               N/A
              tcp_recvspace               N/A
              tcp_mssdflt                 N/A
   ```

2. Set the interface specific values, using either the **ifconfig** or **chdev** command. The **ifconfig** command sets values temporarily, which is recommended for testing. The **chdev** command alters the ODM, so customized values remain valid after reboot.

   – Set the **tcp_recvspace** and **tcp_sendspace** to 64K and enable **tcp_nodelay** by using one of the following:

   ```
   $   ifconfig en0 tcp_recvspace 65536 tcp_sendspace 65536 tcp_nodelay 1

    $   chdev –l en0 –a tcp_recvspace=65536 –a tcp_sendspace=65536 –a
   tcp_nodelay=1
   ```

   – Alternatively, assuming the **no** command reports an **rfc1323=1** global value, the root user can turn **rfc1323** off for all connections over en0 with the following commands:

   ```
   $   ifconfig en0 rfc1323 0
    $   chdev –l en0 –a rfc1323=0
   ```

3. Verify the settings using the **ifconfig** or **lsattr** command, as shown in the following example:

   ```
   $   ifconfig en0
   <UP,BROADCAST,NOTRAILERS,RUNNING,SIMPLEX,MULTICAST,GROUPRT,64BIT>
              en0: flags=e080863
                  inet 9.19.161.100 netmask 0xffffff00 broadcast
   9.19.161.255
                  tcp_sendspace 65536 tcp_recvspace 65536 tcp_nodelay 1
   rfc1323 0
    $   lsattr –El en  0
              rfc1323         0           N/A               True
              tcp_nodelay     1           N/A               True
              tcp_sendspace   65536       N/A               True
              tcp_recvspace   65536       N/A               True
              tcp_mssdflt                 N/A               True
   ```

TCP/IP Addressing

TCP/IP includes an Internet addressing scheme that allows users and applications to identify a specific network or host with which to communicate. An Internet address works like a postal address, allowing data to be routed to the chosen destination. TCP/IP provides standards for assigning addresses to networks, subnetworks, hosts, and sockets, and for using special addresses for broadcasts and local loopback.

Internet addresses are made up of a network address and a host (or local) address. This two–part address allows a sender to specify the network as well as a specific host on the network. A unique, official network address is assigned to each network when it connects to other Internet networks. However, if a local network is not going to connect to other Internet networks, it can be assigned any network address that is convenient for local use.

The Internet addressing scheme consists of Internet Protocol (IP) addresses and two special cases of IP addresses: broadcast addresses and loopback addresses.

Internet Addresses

The Internet Protocol (IP) uses a 32–bit, two–part address field. The 32 bits are divided into four *octets* as in the following:

01111101 00001101 01001001 00001111

These binary numbers translate into:

125 13 73 15

The two parts of an Internet address are the network address portion and the host address portion. This allows a remote host to specify both the remote network and the host on the remote network when sending information. By convention, a host number of 0 is used to refer to the network itself.

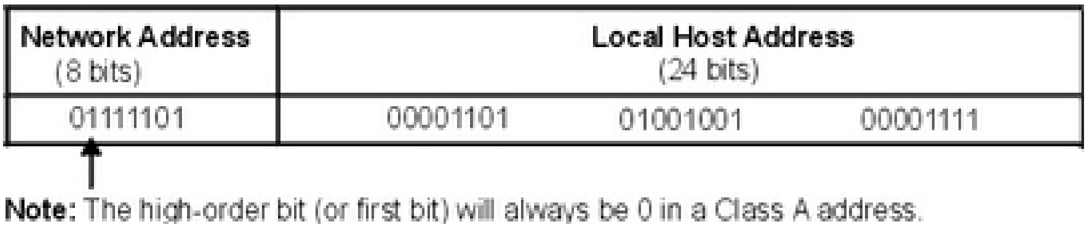
TCP/IP supports three classes of Internet addresses: Class A, Class B, and Class C. The different classes of Internet addresses are designated by how the 32 bits of the address are allocated. The particular address class a network is assigned depends on the size of the network.

Class A Addresses

A Class A address consists of an 8–bit network address and a 24–bit local or host address. The first bit in the network address is dedicated to indicating the network class, leaving 7 bits for the actual network address. Because the highest number that 7 bits can represent in binary is 128, there are 128 possible Class A network addresses. Of the 128 possible network addresses, two are reserved for special cases: the network address 127 is reserved for local loopback addresses, and a network address of all ones indicates a broadcast address.

There are 126 possible Class A network addresses and 16,777,216 possible local host addresses. In a Class A address, the highest order bit is set to 0.

Figure 16. Class A Address This illustration shows a typical class A address structure. The first 8 bits contain the network address (always beginning with a zero). The remaining 24 bits contain the local host address.



Network Address (8 bits)	Local Host Address (24 bits)		
01111101	00001101	01001001	00001111

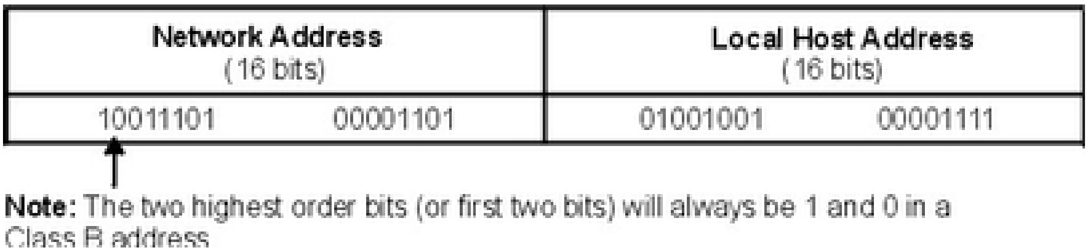
Note: The high-order bit (or first bit) will always be 0 in a Class A address.

The first octet of a Class A address is in the range 1 to 126.

Class B Addresses

A Class B address consists of a 16–bit network address and a 16–bit local or host address. The first two bits in the network address are dedicated to indicating the network class, leaving 14 bits for the actual network address. There are 16,384 possible network addresses and 65,536 local host addresses. In a Class B address, the highest order bits are set to 1 and 0.

Figure 17. Class B Address This illustration shows a typical class B address structure. The first 16 bits contain the network address. The two highest order bits will always be a one and a zero. The remaining 16 bits contain the local host address.



Network Address (16 bits)		Local Host Address (16 bits)	
10011101	00001101	01001001	00001111

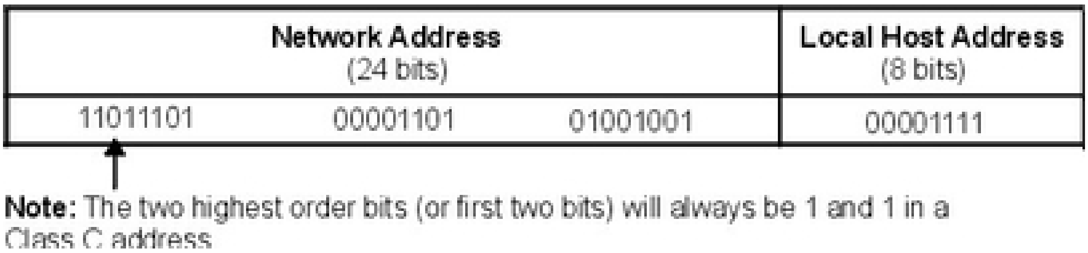
Note: The two highest order bits (or first two bits) will always be 1 and 0 in a Class B address

The first octet of a Class B address is in the range 128 to 191.

Class C Addresses

A Class C address consists of a 24–bit network address and an 8–bit local host address. The first two bits in the network address are dedicated to indicating the network class, leaving 22 bits for the actual network address. Therefore, there are 2,097,152 possible network addresses and 256 possible local host addresses. In a Class C address, the highest order bits are set to 1 and 1.

Figure 18. Class C Address This illustration shows a typical class C address structure. The first 24 bits contain the network address (the two highest order bits will always be a one and a one). The remaining 8 bits contain the local host address.



Network Address (24 bits)			Local Host Address (8 bits)
11011101	00001101	01001001	00001111

Note: The two highest order bits (or first two bits) will always be 1 and 1 in a Class C address

In other words, the first octet of a Class C address is in the range 192 to 223.

When deciding which network address class to use, you must consider how many local hosts there will be on the network and how many subnetworks will be in the organization. If the organization is small and the network will have fewer than 256 hosts, a Class C address is probably sufficient. If the organization is large, then a Class B or Class A address might be more appropriate.

Note:: Class D (1–1–1–0 in the highest order bits) addresses provide for multicast addresses and are supported by UDP/IP under this operating system.

Machines read addresses in binary code. The conventional notation for Internet host addresses is the *dotted decimal*, which divides the 32–bit address into four 8–bit fields. The following binary value:

0001010 00000010 00000000 00110100

can be expressed as:

010.002.000.052 or 10.2.0.52

where the value of each field is specified as a decimal number and the fields are separated by periods.

Note:: The **hostent** command does recognize the following addresses: .08, .008, .09, and .009. Addresses with leading zeros are interpreted as octal, and numerals in octal cannot contain 8s or 9s.

TCP/IP requires a unique Internet address for each network interface (adapter) on a network. These addresses are determined by entries in the configuration database, which must agree with entries in the **/etc/hosts** file or the **named** database if the network is using a name server.

Internet Addresses Using Zeros

When a C class Internet address contains a 0 as the host address portion, (for example, 192.9.200.0), TCP/IP sends a wildcard address on the network. All machines with a Class C address of 192.9.200. *X* (where *X* represents a value between 0 and 254) should respond to the request. This results in a network flooded with requests to nonexistent machines.

Similarly, problems occur for Class B addresses such as 129.5.0.0. All machines with a Class B address of 129.5. *X*. *X*. (where *X* represents a value between 0 and 254) are obliged to respond to the request. In this case, because Class B addresses account for bigger networks than Class C addresses, the network is flooded with significantly more requests to nonexistent machines than for a Class C network.

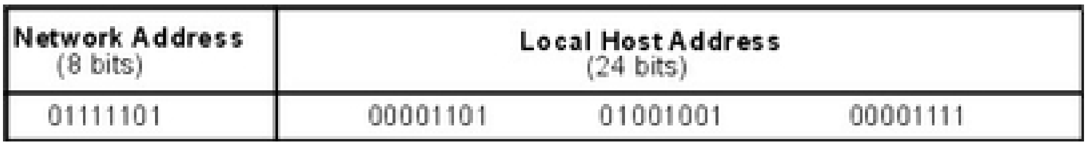
Subnet Addresses

Subnet addressing allows an autonomous system made up of multiple networks to share the same Internet address. The subnetwork capability of TCP/IP also makes it possible to divide a single network into multiple logical networks (subnets). For example, an organization can have a single Internet network address that is known to users outside the organization, yet it can configure its network internally into departmental subnets. In either case, fewer Internet network addresses are required while local routing capabilities are enhanced.

A standard Internet Protocol address field has two parts: a network address and a local address. To make subnets possible, the local address part of an Internet address is divided into a subnet number and a host number. The subnet is identified so that the local autonomous system can route messages reliably.

In the basic Class A Internet address, which consists of an 8–bit network address and 24–bit local address, the local address identifies the specific host machine on the network.

Figure 19. Class A Address This illustration shows a typical class A address structure. The first 8 bits contain the network address (always beginning with a zero). The remaining 24 bits contain the local host address.

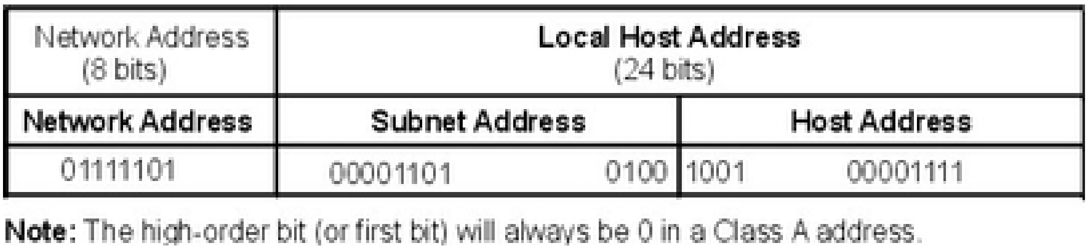


Network Address (8 bits)	Local Host Address (24 bits)		
01111101	00001101	01001001	00001111

To create a subnet address for this Class A Internet address, the local address can be divided into a number identifying the physical network (or subnet) and a number identifying the host on the subnet. Senders route messages to the advertised network address, and the local system takes responsibility for routing messages to its subnets and their hosts. When deciding how to partition the local address into subnet address and host address, you should consider the number of subnets and the number of hosts on those subnets.

In the following figure, the local address is partitioned into a 12–bit subnet address and a 12–bit host address.

Figure 20. Class A Address with Corresponding Subnet Address This illustration shows a typical class A address structure. The first 8 bits contain the network address (always beginning with a zero). The remaining 24 bits contain the local host address with the subnet address occupying the first 8 bits and the host address occupying the last 8 bits.



Network Address (8 bits)	**Local Host Address** (24 bits)	
Network Address	**Subnet Address**	**Host Address**
01111101	00001101 0100	1001 00001111

Note: The high-order bit (or first bit) will always be 0 in a Class A address.

You have flexibility when assigning subnet addresses and host addresses. The bits of the local address can be divided according to the needs and potential growth of the organization and its network structure. The only restrictions are:

- `network_address` is the Internet address for the network.
- `subnet_address` is a field of a constant width for a given network.
- `host_address` is a field that is at least 1–bit wide.

If the width of the `subnet_address` field is 0, the network is not organized into subnets, and addressing to the network is performed using the Internet network address.

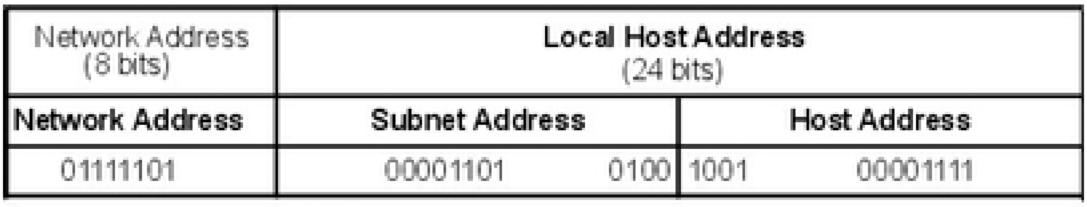
The bits that identify the subnet are specified by a bit mask and, therefore, are not required to be adjacent in the address. However, it is generally desirable for the subnet bits to be contiguous and located as the most significant bits of the local address.

Subnet Masks

When a host sends a message to a destination, the system must determine whether the destination is on the same network as the source or if the destination can be reached directly through one of the local interfaces. The system compares the destination address to the host address using the *subnet mask*. If the destination is not local, the system sends the message on to a gateway. The gateway performs the same comparison to see if the destination address is on a network it can reach locally.

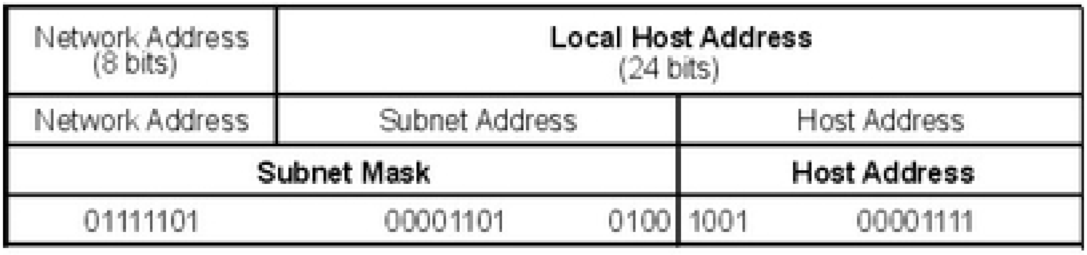
The subnet mask tells the system what the subnet partitioning scheme is. This bit mask consists of the network address portion and subnet address portion of the Internet address.

Figure 21. Class A Address with Corresponding Subnet Address This illustration shows a typical class A address structure. The first 8 bits contain the network address (always beginning with a zero). The remaining 24 bits contain the local host address with the subnet address occupying the first 8 bits and the host address occupying the last 8 bits.



Network Address (8 bits)	Local Host Address (24 bits)	
Network Address	**Subnet Address**	**Host Address**
01111101	00001101 0100	1001 00001111

Class A Address with Corresponding Subnet Address



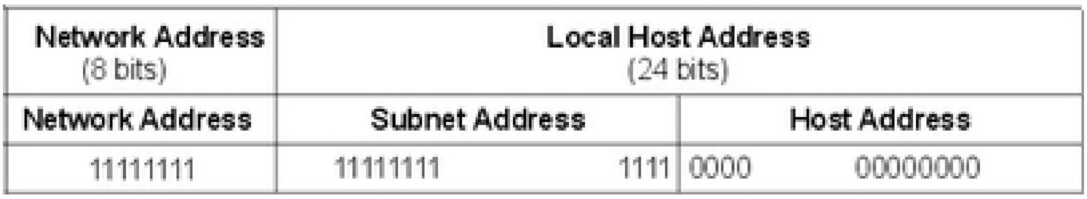
Network Address (8 bits)	Local Host Address (24 bits)	
Network Address	Subnet Address	Host Address
Subnet Mask		**Host Address**
01111101	00001101 0100	1001 00001111

Class A Address with Corresponding Subnet Mask

For example, the subnet mask of the Class A address with the partitioning scheme defined above is shown in this figure.

The subnet mask is a set of 4 bytes, just like the Internet address. The subnet mask consists of high bits (1’s) corresponding to the bit positions of the network and subnetwork address, and low bits (0’s) corresponding to the bit positions of the host address. A subnet mask for the previous address looks like the following figure.

Figure 22. Example Subnet Mask This illustration shows a an example of a subnet mask structure. The first 8 bits contain the network address. The remaining 24 bits contain the local host address with the subnet address occupying the first 8 bits and the host address occupying the last 8 bits.



Network Address (8 bits)	Local Host Address (24 bits)	
Network Address	**Subnet Address**	**Host Address**
11111111	11111111 1111	0000 00000000

Address Comparison

The destination address and the local network address are compared by performing the logical AND and exclusive OR on the subnet mask of the source host.

The comparison process is outlined below:

1. Perform a logical AND of the destination address and the mask of the local subnet address.
2. Perform an exclusive OR on the result of the previous operation and the local net address of the local interface.

   If the result is all 0’s, the destination is assumed to be reachable directly through one of the local interfaces.
3. If an autonomous system has more than one interface (therefore more than one Internet address), the comparison process is repeated for each local interface.

For example, assume that there are two local interfaces defined for a host network, T125. Their Internet addresses and the binary representations of those addresses are shown in the following example:

Local Network Interface Addresses

```
CLASS A   73.1.5.2    =  01001001 00000001 00000101 00000010


 CLASS B   145.21.6.3  =  10010001 00010101 00000110 00000011
```

The corresponding subnet masks for the local network interfaces are shown in the following example:

Local Network Interface Addresses

```
CLASS A   73.1.5.2    =  11111111 11111111 11100000 00000000


 CLASS B   145.21.6.3  =  11111111 11111111 11111111 11000000
```

If the source network, T125, is requested to send a message to a destination network with the host address 114.16.23.8 (represented in binary as: 01110010 00010000 00010111 00001000), the system checks whether the destination can be reached through a local interface.

Note:: The **subnetmask** keyword must be set in the configuration database of each host that is to support subnets. Before the subnetwork capability can be used, all hosts on the network must support it. Set the subnet mask permanently in the configuration database using the Web-based System Manager Network application or the Network Interface Selection menu in SMIT. The subnet mask can also be set in the running system using the **ifconfig** command. Using the **ifconfig** command to set the subnet mask is not a permanent change.

Broadcast Addresses

The TCP/IP can send data to all hosts on a local network or to all hosts on all directly connected networks. Such transmissions are called *broadcast messages*. For example, the **routed** routing daemon uses broadcast messages to query and respond to routing queries.

For data to be broadcast to all hosts on all directly connected networks, User Datagram Protocol (UDP) and Internet Protocol (IP) are used to send the data, and the host destination address in the IP header has all bits set to 1. For data to be broadcast to all hosts on a specific network, all the bits in the local address part of the IP address are set to 0. There are no user commands that use the broadcast capability, although such commands, or programs, can be developed.

The broadcast address can be changed temporarily by changing the *broadcast* parameter in the **ifconfig** command. Change the broadcast address permanently by using the Web–based System Manager, **wsm**, or the SMIT fast path **smit chinet**. Changing the broadcast address may be useful if you need to be compatible with older versions of software that use a different broadcast address; for example, the host IDs are all set to 0.

Local Loopback Addresses

The Internet Protocol defines the special network address, 127.0.0.1, as a local loopback address. Hosts use local loopback addresses to send messages to themselves. The local loopback address is set by the configuration manager during the system startup process. Local loopback is implemented in the kernel and can also be set with the **ifconfig** command. Loopback is invoked when the system is started.

TCP/IP Name Resolution

Although 32–bit Internet addresses provide machines an efficient means of identifying the source and destination of datagrams sent across an internetwork, users prefer meaningful, easily remembered names. Transmission Control Protocol/Internet Protocol (TCP/IP) provides a naming system that supports both flat and hierarchical network organizations.

The topics discussed in this section are:

Naming

Naming in flat networks is very simple. Host names consist of a single set of characters and generally are administered locally. In flat TCP/IP networks, each machine on the network has a file (**/etc/hosts**) containing the name–to–Internet–address mapping information for every host on the network. The administrative burden of keeping each machine naming file current grows as the TCP/IP network grows. When TCP/IP networks become very large, as on the Internet, naming is divided hierarchically. Typically, the divisions follow the network organization. In TCP/IP, hierarchical naming is known as the *domain name system* (DNS) and uses the DOMAIN protocol. The DOMAIN protocol is implemented by the **named** daemon in TCP/IP.

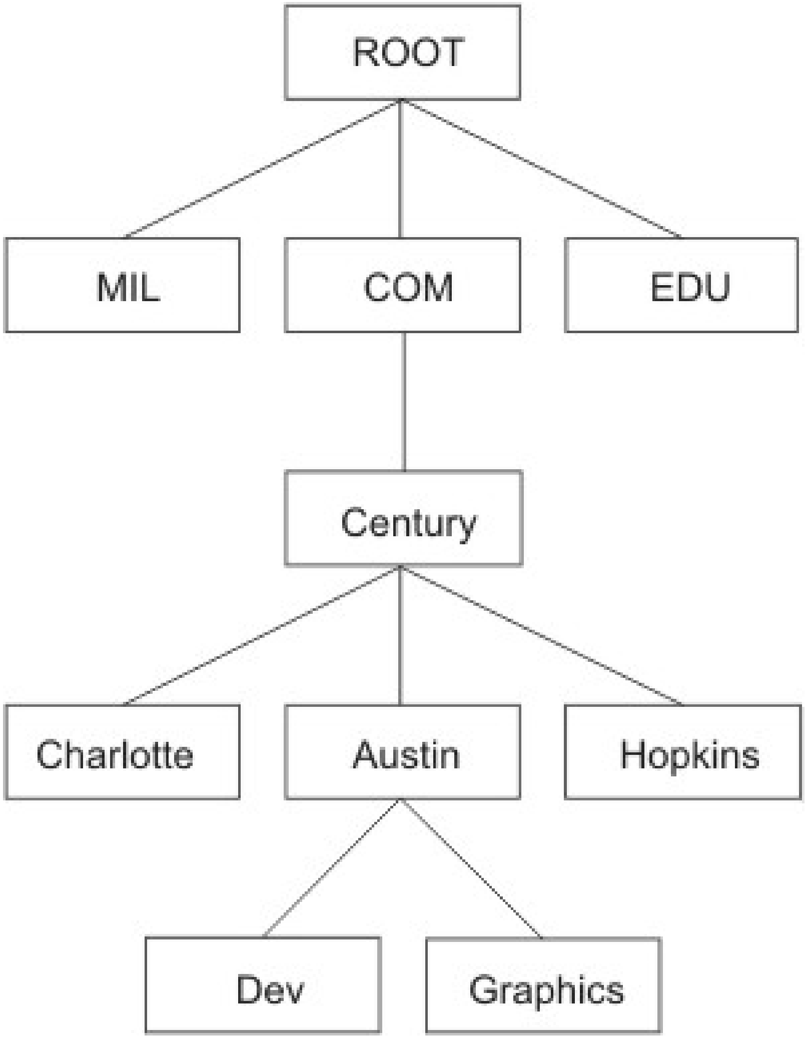
As in naming for flat networks, the domain name hierarchy provides for the assignment of symbolic names to networks and hosts that are meaningful and easy for users to remember. However, instead of each machine on the network keeping a file containing the name–to–address mapping for all other hosts on the network, one or more hosts are selected to function as *name servers.* Name servers translate (resolve) symbolic names assigned to networks and hosts into the efficient Internet addresses used by machines. A name server has complete information about some part of the domain, referred to as a *zone*, and it has *authority* for its zone.

Naming Authority

In a flat network, all hosts in the network are administered by one central authority. This form of network requires that all hosts in the network have unique host names. In a large network, this requirement creates a large administrative burden on the central authority.

In a domain network, groups of hosts are administered separately within a tree–structured hierarchy of domains and subdomains. In this case, host names need to be unique only within the local domain, and only the *root domain* is administered by a central authority. This structure allows subdomains to be administered locally and reduces the burden on the central authority. For example, the root domain of the Internet consists of such domains as `com` (commercial organizations), `edu` (educational organizations), `gov` (governmental organizations), and `mil` (military groups). New top–level domains can only be added by the central authority. Naming at the second level is delegated to designated agents within the respective domains. For example, in the following figure, `com` has naming authority for all commercial organization subdomains beneath it. Likewise, naming at the third level (and so on) is delegated to agents within that level. For example, in the Domain Structure of the Internet figure, Century has naming authority for its subdomains Austin, Hopkins, and Charlotte.

Figure 23. Domain Structure of the Internet This figure illustrates the hierarchical structure of the internet. It begins at the top with the root and branches to the next level containing the mil, com, and edu domains. Below the com domain is another level containing Charlotte, Austin, and Hopkins. Below Austin is Dev and Graphics.



Century's Austin subdomain might also be divided into zones, for example, Dev and Graphics. In this case, the zone `austin.century.com` has all the data contained in the domain `austin.century.com`, except that which was delegated to Dev and Graphics. The zone `dev.century.com` would contain only the data delegated to Dev; it would know nothing about Graphics, for example. The zone `austin.century.com` (as opposed to the domain of the same name) would contain only that data not delegated to other zones.

Naming Conventions

In the hierarchical domain name system, names consist of a sequence of case–insensitive subnames separated by periods with no embedded blanks. The DOMAIN protocol specifies that a local domain name must be fewer than 64 characters and that a host name must be fewer than 32 characters in length. The host name is given first, followed by a period (.), a series of local domain names separated by periods, and finally the root domain. A fully specified domain name for a host, including periods, must be fewer than 255 characters in length and in the following form:

```
host.subdomain1.[subdomain2 . . . subdomain].rootdomain
```

Since host names must be unique within a domain, you can use an abbreviated name when sending messages to a host within the same domain. For example, instead of sending a message to `smith.eng.lsu.edu`, a host in the `eng` domain could send a message to `smith`. Additionally, each host can have several aliases that other hosts can use when sending messages.

Choosing Names for the Hosts on Your Network

The purpose of using names for hosts is to provide a quick, easy, and unambiguous way to refer to the computers in your network. Internet system administrators have discovered that there are good, as well as poor, choices for host names. These suggestions are intended to help you avoid common pitfalls in choosing host names.

The following are some suggestions for choosing unambiguous, easy to remember host names:

- Terms that are rarely used, for example, `sphinx` or `eclipse`.
- Theme names, such as colors, elements (for example, `helium`, `argon`, or `zinc`), flowers, fish, and others.
- Real words (as opposed to random strings of characters).

The following are some examples of poor choices. In general, these are poor choices because they are difficult to remember or are confusing (either to humans or computers):

- Terms that are already in common use, for example, `up`, `down`, or `crash`.
- Names containing only numbers.
- Names that contain punctuation marks.
- Names that rely on case distinction, for example, `Orange` and `orange`.
- The name or initials of the primary user of the system.
- Names having more than 8 characters.
- Unusual or purposefully incorrect spellings, for example, `czek`, which could be confused with ”check” or ”czech.”
- Names that are, or resemble, domain names, for example, `yale.edu`.

Name Servers

In a flat name space, all names must be kept in the **/etc/hosts** file on each host on the network. If the network is very large, this can become a burden on the resources of each machine.

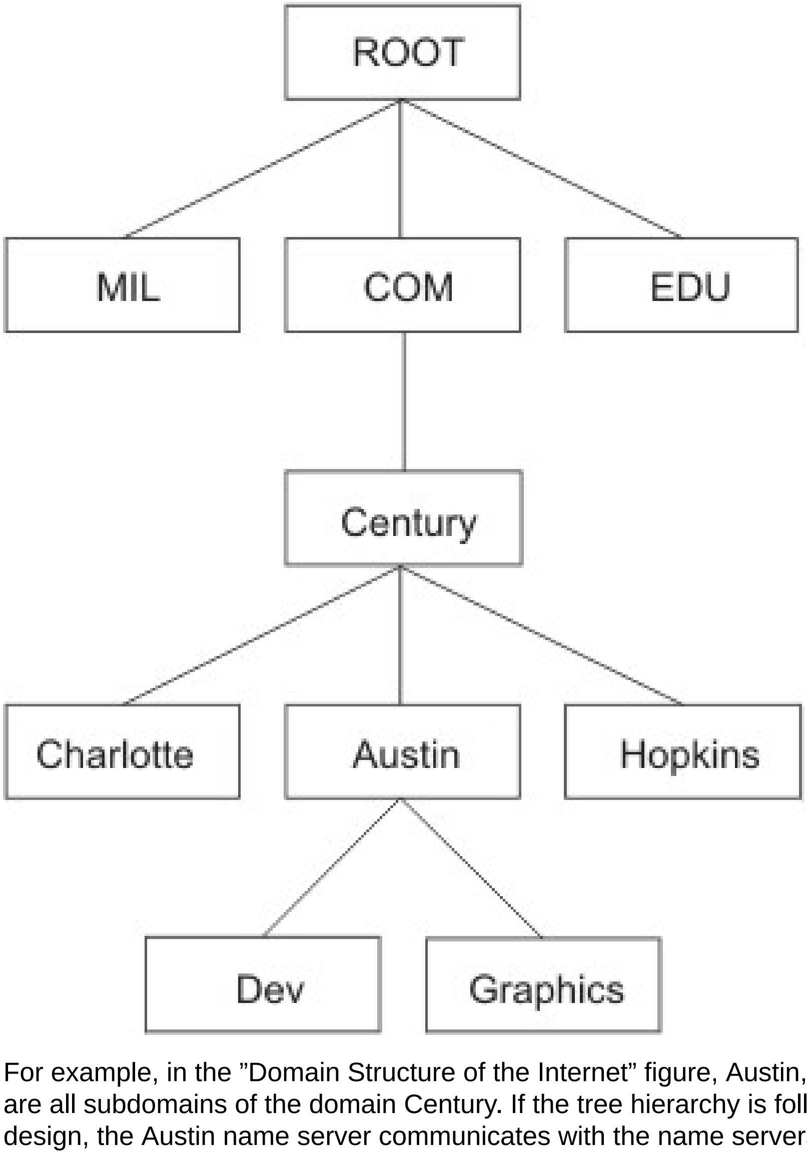
In a hierarchical network, certain hosts designated as *name servers* resolve names into Internet addresses for other hosts. This has two advantages over the flat name space. It keeps the resources of each host on the network from being tied up in resolving names, and it keeps the person who manages the system from having to maintain name resolution files on each machine on the network. The set of names managed by a single name server is known as its *zone of authority*.

Note:: Although the host machine that performs the name resolution function for a zone of authority is commonly referred to as a *name server* host, the process controlling the function, the **named** daemon, is the actual name server process.

To further reduce unnecessary network activity, all name servers *cache* (store for a period of time) name–to–address mappings. When a client asks a server to resolve a name, the server checks its cache first to see if the name has been resolved recently. Because domain and host names do change, each item remains in the cache for a limited length of time specified by the TTL of the record. In this way, authorities can specify how long they expect the name resolution to be accurate.

Within any autonomous system there can be multiple name servers. Typically, name servers are organized hierarchically and correspond to the network organization. Referring to the ”Domain Structure of the Internet” figure, each domain might have a name server responsible for all subdomains within the domain. Each subdomain name server communicates with the name server of the domain above it (called the *parent* name server), as well as with the name servers of other subdomains.

Figure 24. Domain Structure of the Internet This figure illustrates the hierarchical structure of the internet. It begins at the top with the root and branches to the next level containing the mil, com, and edu domains. Below the com domain is another level containing Charlotte, Austin, and Hopkins. Below Austin is Dev and Graphics.



For example, in the ”Domain Structure of the Internet” figure, Austin, Hopkins, and Charlotte are all subdomains of the domain Century. If the tree hierarchy is followed in the network design, the Austin name server communicates with the name servers of Charlotte and

Hopkins as well as with the parent Century name server. The Austin name server also communicates with the name servers responsible for its subdomains.

There are several types of name servers:

Master Name Server
Loads its data from a file or disk and can delegate authority to other servers in its domain.

Slave Name Server
Receives its information at system startup time for the given zone of authority from a master name server, and then periodically asks the master server to update its information. On expiration of the refresh value in the start of authority (SOA) Resource Record on a slave name server, or on receipt of a Notify message from the master name server, the slave reloads the database from the master if the serial number of the database on the master is greater than the serial number in the current database on the slave. If it becomes necessary to force a new zone transfer from the master, simply remove the existing slave databases and refresh the **named** daemon on the slave name server.

Stub Name Server
Although its method of database replication is similar to that of the slave name server, the stub name server only replicates the name server records of the master database rather than the whole database.

Hint Server
Indicates a name server that relies only on the hints that it has built from previous queries to other name servers. The hint name server responds to queries by asking other servers that have the authority to provide the information needed if a hint name server does not have a name–to–address mapping in its cache.

Forwarder or Client Server
Forwards queries it cannot satisfy locally to a fixed list of forwarding servers. Forwarding–only servers (a forwarder that obtains information and passes it on to other clients, but that is not actually a server) does not interact with the master name servers for the root domain and other domains. The queries to the forwarding servers are recursive. There can be one or more forwarding servers, which are tried in turn until the list is exhausted. A client and forwarder configuration is typically used when you do not want all the servers at a given site to interact with the rest of the Internet servers, or when you want to build a large cache on a select number of name servers.

Remote Server
Runs all the network programs that use the name server without the name server process running on the local host. All queries are serviced by a name server that is running on another machine on the network.

One name server host can perform in different capacities for different zones of authority. For example, a single name server host can be a master name server for one zone and a slave name server for another zone.

Name Resolution

The process of obtaining an Internet address from a host name is known as name resolution and is done by the **gethostbyname** subroutine. The process of translating an Internet address into a host name is known as reverse name resolution and is done by the **gethostbyaddr** subroutine. These routines are essentially accessors into a library of name translation routines known as *resolvers*.

Resolver routines on hosts running TCP/IP normally attempt to resolve names using the following sources:

1. BIND/DNS (named)
2. Network Information Service (NIS)
3. Local **/etc/hosts** file

When NIS+ is installed, lookup preferences are set using the **irs.conf** file. For more information, see *AIX 5L Version 5.2* Network Information Service *(NIS and NIS+) Guide*.

To resolve a name in a domain network, the resolver routine first queries the domain name server database, which might be local if the host is a domain name server or on a foreign host. Name servers translate domain names into Internet addresses. The group of names for which a name server is responsible is its zone of authority. If the resolver routine is using a remote name server, the routine uses the domain name protocol (DOMAIN) to query for the mapping. To resolve a name in a flat network, the resolver routine checks for an entry in the local **/etc/hosts** file. When NIS or NIS+ is used, the **/etc/hosts** file on the master server is checked.

By default, resolver routines attempt to resolve names using the above resources. BIND/DNS is tried first. If the **/etc/resolv.conf** file does not exist or if BIND/DNS could not find the name, NIS is queried if it is running. NIS is authoritative over the local **/etc/hosts**, so the search ends here if it is running. If NIS is not running, then the local **/etc/hosts** file is searched. If none of these services can find the name, then the resolver routines return with HOST_NOT_FOUND. If all of the services are unavailable, then the resolver routines return with SERVICE_UNAVAILABLE.

The default order described above can be overwritten by creating the **/etc/irs.conf** configuration file and specifying the desired order. Also, both the default and **/etc/irs.conf** orderings can be overwritten with the environment variable, **NSORDER**. If either the **/etc/irs.conf** file or **NSORDER** environment variable are defined, then at least one value must be specified along with the option.

To specify host ordering with the **/etc/irs.conf** file:

```
hosts   value   [ continue ]
```

The order is specified with each method indicated on a line by itself. The *value* is one of the listed methods and the `continue` keyword indicates that another resolver method follows on the next line.

To specify host ordering with the **NSORDER** environment variable:

```
NSORDER=   value ,   value ,   value
```

The order is specified on one line with values separated by commas. White spaces are permitted between the commas and the equal sign.

For example, if the local network is organized as a flat network, then only the **/etc/hosts** file is needed. Given this example, the **/etc/irs.conf** file contains the following line:

```
hosts local
```

Alternatively, the **NSORDER** environment variable can be set as:

```
NSORDER=local
```

If the local network is a domain network using a name server for name resolution and an **/etc/hosts** file for backup, then both services should be specified. Given this example, the **/etc/irs.conf** file contains the following lines:

```
hosts dns continue
 hosts local
```

The **NSORDER** environment variable is set as:

```
NSORDER=bind,local
```

Note:: The values listed must be in lowercase.

When following any defined or default resolver ordering, the search algorithm continues from one resolver to the next only if:

- The current service is not running, therefore, it is unavailable.
- The current service cannot find the name and is not authoritative.

If the **/etc/resolv.conf** file does not exist, then BIND/DNS is considered not set up or running, and therefore it is not available. If the **getdomainname** and **yp_bind** subroutines fail, then the NIS service is considered not set up or running, and therefore it is not available. If the **/etc/hosts** file could not be opened, then a local search is impossible, and therefore the file and service are unavailable.

When a service is listed as *authoritative*, it means that this service is the expert of its successors and has all pertinent names and addresses. Resolver routines do not try successor services, because successors might contain only a subset of the information in the authoritative service. Name resolution ends at service listed as authoritative, even if it does not find the name (in which case, the resolver routine returns HOST_NOT_FOUND). If an authoritative service is not available, then the next service specified is queried.

An authoritative source is specified with the string `=auth` directly behind a value. The entire word, `authoritative` can be typed in, but only the `auth` string is used. For example, if the **NSORDER** environment variable contains the following:

```
hosts = nis=auth,dns,local
```

The search ends after the NIS query (if NIS is running), regardless of whether the name was found. If NIS is not running, then the next source is queried, which is DNS.

TCP/IP name servers use caching to reduce the cost of searching for names of hosts on remote networks. Instead of searching for a host name each time a request is made, a name server first looks at its cache to see if the host name has been resolved recently. Since domain and host names do change, each item remains in the cache for a limited length of time specified by the time–to–live (TTL) value of the record. In this way, name servers can specify how long they expect their responses to be considered authoritative.

Potential Host Name Conflict Between name server and sendmail

In a DNS environment, a host name that is set using the **hostname** command from the command line or in the **rc.net** file format must be the official name of the host as returned by the name server. Generally, this name is the full domain name of the host in the form:

```
host.subdomain.subdomain.rootdomain
```

Note:: Resolver routines require the default domain to be set. If the default domain is not set in the **hostname** command, then it must be set in the **/etc/resolv.conf** file.

If the host name is not set up as a fully qualified domain name, and if the system is set up to use a domain name server in conjunction with the **sendmail** program, the **sendmail** configuration file (**/etc/sendmail.cf**) must be edited to reflect this official host name. In addition, the domain name macros in this configuration file must be set for the **sendmail** program to operate correctly.

Note:: The domain specified in the **/etc/sendmail.cf** file takes precedence over the domain set by the **hostname** command for all **sendmail** functions.

Potential Domain Name Conflict Between name server and sendmail

For a host that is in a DOMAIN network but is not a name server, the local domain name and domain name server are specified in the **/etc/resolv.conf** file. In a DOMAIN name server host, the local domain and other name servers are defined in files read by the **named** daemon when it starts.

Reverse Address Resolution Protocol

The Reverse Address Resolution Protocol (RARP) translates unique hardware addresses into Internet addresses on the Ethernet local area network (LAN) adapter (Ethernet protocol only). Standard Ethernet protocol is supported with the following restrictions:

- The server only replies to RARP requests.
- The server only uses permanent ARP table entries.
- The server does not use dynamic ARP table entries.
- The server does not automatically reply for itself.

The system administrator must manually build and maintain a table of permanent ARP entries using the **arp** command. A specific ARP table entry must be added on the server for each host that requires RARP replies from an authoritative source.

Performing Local Name Resolution (/etc/hosts)

Configure the **/etc/hosts** file if your network is small, and you are using a flat naming scheme. Even if you are using a hierarchical (or domain) naming scheme with name servers, you might want to configure the **/etc/hosts** file to identify hosts that are not known by the name servers.

Configure your system for local host resolution using the Web–based System Manager, the System Management Interface Tool (SMIT), or commands. If you choose the command method, be sure to preserve the format of the **/etc/hosts** file, as described in Hosts File Format for TCP/IP in the *AIX 5L Version 5.2 Files Reference*.

Local name resolution tasks

Task	**SMIT fast path**	**Command or file**	Web-based System Manager Management Environment
List All the Hosts	**smit lshostent**	view **/etc/hosts**	Software —> **Network** —> **TCPIP (IPv4 and IPv6)** —> **TCPIP Protocol Configuration** —> **TCP/IP** —> **Configure TCP/IP** —> **Advanced Methods** —> **Hosts File** —> **Contents of /etc/hosts file**.

Add a Host	**smit mkhostent**	edit **/etc/hosts**	Software —> **Network** —> **TCPIP (IPv4 and IPv6)** —> **TCPIP Protocol Configuration** —> **TCP/IP** —> **Configure TCP/IP** —> **Advanced Methods** —> **Hosts File**. In **Add/Change host entry**, complete the following fields: **IP Addresses**, **Host name**, **Alias(es)**, and **Comment**. Click **Add/Change Entry** —> **OK**.
Change/Show Characteristics of a Host	**smit chhostent**	edit **/etc/hosts**	Software —> **Network** —> **TCPIP (IPv4 and IPv6)** —> **TCPIP Protocol Configuration** —> **TCP/IP** —> **Configure TCP/IP** —> **Advanced Methods** —> **Hosts File**. Select a host in **Contents of /etc/hosts/file**, and change data in **Add/Change host entry**. Click **Add/Change Entry** —> **OK**.
Remove a Host	**smit rmhostent**	edit **/etc/hosts**	Software —> **Network** —> **TCPIP (IPv4 and IPv6)** —> **TCPIP Protocol Configuration** —> **TCP/IP** —> **Configure TCP/IP** —> **Advanced Methods** —> **Hosts File**. Select a host in **Contents of /etc/hosts/file**, and click **Delete Entry** —> **OK**.

Planning for DOMAIN Name Resolution

If you are part of a larger internetwork, coordinate setting up your domain and name servers with the central authority.

The following suggestions can help you plan your own DOMAIN name resolution system:

- Because of the vast possibilities in architecture and configuration, become familiar with TCP/IP, DNS, and BIND before you solidify any plans. If you plan to use a network information service, become familiar with NFS and NIS as well. Books about these topics are widely available. For more information about NIS and NIS+, see *AIX 5L Version 5.2* Network Information Service *(NIS and NIS+) Guide*.
- Plan ahead.

  >Changing a name is *much* more difficult than setting up the initial one. Obtain consensus from your organization on network, gateway, name server, and host names before you set up your files.
- Set up redundant name servers.

  If you cannot set up redundant name servers, be sure to set up slave and hint name servers so you have some type of backup.
- In selecting the name servers, keep the following in mind:
  - Choose machines that are physically closest to exterior systems.
  - The name servers should be as independent as possible. Try for different power supplies and independent cabling.
  - Find another network to back up your name resolution service, and do the same for other networks.
- Test the servers.
  - Test both regular and reverse name resolution.
  - Test zone transfer from master to slave name servers.
  - Test each name server after a system crash and reboot.
- Send name resolution requests to forwarder servers before they go to exterior name servers. This allows your name servers to share caches and improve performance by reducing the load on your master name servers.

```
objectclass container
         requires
                 objectclass,
                 cn
 objectclass hosts
         requires
                 objectclass,
                 hname
         allows
                 addr
                 halias,
                 comment
```

Name Server Overview

In a hierarchical network, certain hosts are designated as *name servers*. These hosts resolve names into IP addresses for other hosts. The **named** daemon controls the name server function and, therefore, must be run on a name server host.

Before you configure a name server, decide which type or types best fit the network it will serve. There are several types of name servers.

A *master name server* actually stores the database containing name–to–address mapping information. It loads its data from a file or disk and can delegate authority to other servers in its domain. A *slave name server* or *stub name server* receives its information at system startup time for a given zone of authority from a master name server, and then periodically asks the master server to update its information. A *hint name server* responds to requests to resolve names by querying other servers that have the authority to provide the information needed.

Note:: Previous generations of the **named** name server specified the master name server as the primary name server, the slave name server as the secondary name server, and the hint name server as the caching–only name server. Any reference to the **named.conf** file in this documentation is specific to AIX 4.3.2 and later versions.

Keep in mind that a name server can function in different capacities for different zones of authority. For example, one name server host can be a master name server for one zone and a slave name server for another zone. If your system has NIS or NIS+ installed, these services can also provide name resolution. For more information, see *AIX 5L Version 5.2* Network Information Service *(NIS and NIS+) Guide*.

There are several files involved in configuring name servers.

conf This file is read when the **named** daemon starts. The records in the **conf** file tell the **named** daemon which type of server it is, which domains it has authority over (its zones of authority), and where to get the data for initially setting up its database. The default name of this file is **/etc/named.conf**. However, you can change the name of this file by specifying the name and path of the file on the command line when the **named** daemon is started. If you intend to use the **/etc/named.conf** as the **conf** file and it does not exist, a message is generated in **syslog** file and **named** terminates. However, if an alternative **conf** file is specified, and the alternative file does not exist, an error message is not generated and **named** continues.

cache Contains information about the local cache. The local cache file contains the names and addresses of the highest authority name servers in the network. The cache file uses the Standard Resource Record Format. The name of the cache file is set in the **conf** file.

domain data

There are three typical domain data files, also referred to as the **named** data files. The **named** *local* file contains the address resolution information for local loopback. The **named** *data* file contains the address resolution data for all machines in the name server zone of authority. The **named** *reverse data* file contains the reverse address resolution information for all machines in the name server zone of authority. The domain data files use the Standard Resource Record Format. Their file names are user definable and are set in the **conf** file. By convention, the names of these files generally include the name of the daemon (**named**), and the type of file and name of the domain is given in the extension. For example, the name server for the domain abc might have the following files:

```
named.abc.data
named.abc.rev
named.abc.local
```

When modifying the **named** data files the serial number in the SOA Resource Record must be incremented for slave name servers to properly realize the new zone changes.

resolv.conf

The presence of this file indicates to a host to go to a name server to resolve a name first. If the **resolv.conf** file does not exist, the host looks in the **/etc/hosts** file for name resolution. On a name server, the **resolv.conf** file must exist and can contain the local host address, the loopback address (127.0.0.1), or be empty.

Note::

The resolver routines require the default domain be set. If the default domain is not set in the **/etc/resolv.conf** file, then it must be set in the **hostname**

Time–to–live (TTL) is specified in resource records. If TTL is not specified in a record, the length of this time period defaults to the minimum field as defined in the start of authority (SOA) record for that zone. TTL is used when data is stored outside a zone (in a cache) to ensure that the data is not retained indefinitely.

Configuring Name Servers

For procedures on how to configure master, slave, and hint name servers, see Configure Domain Name Servers.

Configuring a Domain Mail Server

Configuring a domain mail server provides users external to your organization a simple method for addressing mail to your users. That is, without a domain mail server, the mail address must specify a particular host in your organization. For example sam@orange.widget.com, where widget.com is your organization's domain name, and orange is the host that sam uses. But with a domain mail server, users outside your organization can simply specify the user name and domain name, without having to know which host the user uses, for example, sam@widget.com.

To configure a domain mail server, use the Web–based System Manager, **wsm**, or use one of the following procedures.

To Configure a Domain Mail Server

1. Create a mail exchanger (MX) record and an address (A) record for the mail server black.widget.com:

```
widget.com           IN     MX      10 black.widget.com
 widget.com           IN      A      192.10.143.9
 black.widget.com     IN      A      192.10.143.9
```

2. Edit **sendmail.cf** on the mail server (`black.widget.com`) to add the domain alias (the **w** class):

   ```
   Cw $w $?D$w.$D$. widget.com
   ```

3. Mail clients must know where to send their non–local mail, so edit **sendmail.cf** on each client to point to the mail server (the **S** macro):

   ```
   DRblack.widget.com
   ```

4. Use the **NameServOpt** option to configure the **sendmail** daemon so everyone can use the MX records defined in the name server `brown.widget.com`.
5. Add aliases for users in the domain that do not have accounts on the mail server using the aliases file, for example:

   ```
   sam:sam@orange.widget.com
    david:david@green.widget.com
    judy:judy@red.widget.com
   ```

   Note:: Mailbox (MB) records can serve the same function.

6. The serial number in the SOA Resource Record must be incremented because the database has been modified.
7. Refresh the name server database by issuing the **refresh –s named** command.
8. On the clients, run the **refresh –s sendmail** command to make the changes take effect.

There are other methods to configure a domain mail server. The following procedures use mailbox (MB), mail rename (MR), and mail group (MG) records.

To Configure a Domain Mail Server Using mailbox (MB) Records

1. Define a mailbox (MB) record for each user in the domain. Add entries such as:

   ```
   sam IN MB orange.widget.com.
   ```

   to the **/usr/local/domain/named.abc.data** file on host `brown.widget.com`. These entries identify to the mail server `black.widget.com` where to send mail for each user in the domain.

2. Configure the **sendmail** daemon on the mail server `black.widget.com` to use the MB records defined in the name server `brown.widget.com`. Use the **NameServOpt** option.
3. Increment the serial number in the SOA Resource Record, because the database has been modified.
4. Refresh the name server database by running the **refresh –s named** command.
5. Type the **refresh –s sendmail** command to make the changes take effect.

Defining a Mail Rename (MR) Record for a User

1. Edit the **/usr/local/domain/named.abc.data** file on your domain name server.
2. Add a Mail Rename record for each alias. For example, if a user `sam` has an alias `sammy`, the Mail Rename record is:

   ```
   sammy IN MR sam
   ```

   This record causes all mail addressed to `sammy` to be delivered to `sam`. Each MR record should be entered on a line by itself.

3. The serial number in the SOA Resource Record must be incremented, because the database has been modified.
4. Refresh the name server database by typing the **refresh –s named** command.
5. Type the **refresh –s sendmail** command to make the changes take effect.

Defining Mail Group (MG) Member Records

1. Edit the **/usr/local/domain/named.abc.data** file on your domain name server.
2. Add MG records for each mail group. MG records function like the **/etc/aliases** file, with the aliases maintained on the name server. For example:

   ```
   users IN HINFO users–request widget.com
   users IN MG sam
   users IN MG david
   users IN MG judy
   ```

   This example causes all mail addressed to `users@widget.com` to be delivered to `sam`, `david`, and `judy`. Enter each MG record on a line by itself.

   Note:: Users `sam`, `david`, and `judy` must have MB records defined.
3. The serial number in the SOA Resource Record must be incremented, because the database has been modified.
4. Refresh the name server database by typing the **refresh –s named** command.
5. Type the **refresh –s sendmail** command to make the changes take effect.

Defining Mail Exchanger (MX) Records

1. Edit the **/usr/local/domain/named.abc.data** file on your domain name server.
2. Add MX records for each machine not directly connected to your network to which you wish to forward mail. For example, if mail addressed to users on `purple.widget.com` should be forwarded to `post.office.widget`, the MX record looks similar to the following:

   ```
   purple.widget.com IN MX 0 post.office.widget.
   ```

   You must specify both host and machine names when using MX records. Enter each MG record on a line by itself. You can use wildcards, for example:

   ```
   *.widget.com IN MX 0 post.office.widget.
   ```

   This example causes mail to an unknown host (a host without an explicit MX record) in the `widget.com` domain to be forwarded to `post.office.widget`.

   Note:: Wildcard MX records are not appropriate for use on the Internet.
3. The serial number in the SOA Resource Record must be incremented because the database has been modified.
4. Refresh the name server database by typing the **refresh –s named** command.
5. Type the **refresh –s sendmail** command to make the changes take effect.

Configuring a Forwarder

To configure a forwarder server, use the Web–based System Manager, **wsm**, or use the following procedure, which edits a series of files and then uses SMIT or the command line to start the **named** daemon.

1. Edit the **/etc/named.conf** file. If there is no **named.conf** file in the **/etc** directory, copy the **/usr/samples/tcpip/named.conf** sample file into the **/etc** directory and edit it. See the ”named.conf File Format for TCP/IP” in the *AIX 5L Version 5.2 Files Reference* for more information and a detailed example of a conf file.
   - Specify a forwarders line in the options stanza of the **/etc/named.conf** file that lists the IP addresses of the name servers that should receive the forwarded requests. For example:

     ```
     options {
          ...
          directory ”/usr/local/domain”;
          forwarders { 192.100.61.1; 129.35.128.222; };
          ...
      };
     ```

– Specify the loopback zone. For example:

```
zone ”0.0.127.in–addr.arpa” in {
     type master;
     file ”named.abc.local”;
 };
```

– Specify the hint zone. For example:

```
zone ”.” IN {
     type hint;
     file ”named.ca”;
 };
```

2. Edit the **/usr/local/domain/named.ca** file. See the ”DOMAIN Cache File Format for TCP/IP” in the *AIX 5L Version 5.2 Files Reference* for more information and a detailed example of a cache file.

   This file contains the addresses of the servers that are authoritative name servers for the root domain of the network. For example:

   ```
   ; root name servers.
   .          IN    NS    relay.century.com.
    relay.century.com.   3600000    IN    A     129.114.1.2
   ```

   Note:: All lines in this file must be in Standard Resource Record Format.

3. Edit the **/usr/local/domain/named.abc.local** file. See the DOMAIN Local Data File Format for TCP/IP in the *AIX 5L Version 5.2 Files Reference* for more information and a detailed example of a local data file.

   a. Specify the start of authority (SOA) of the zone and the default time–to–live information. For example:

   ```
   $TTL 3h    ;3 hour

    @ IN SOA venus.abc.aus.century.com. gail.zeus.abc.aus.century.com.  (

                                1       ;serial
                                3600    ;refresh
                                600     ;retry
                                3600000 ;expire
                                86400   ;negative caching TTL
    )
   ```

   b. Specify the name server (NS) record. For example:

   ```
    <tab>      IN    NS     venus.abc.aus.century.com.
   ```

   c. Specify the pointer (PTR) record.

   ```
   1      IN    PTR    localhost.
   ```

   Note:: All lines in this file must be in Standard Resource Record Format.

4. Create an **/etc/resolv.conf** file by typing the following command:

   ```
   touch /etc/resolv.conf
   ```

   The presence of this file indicates that the host should use a name server, not the **/etc/hosts** file, for name resolution.

   Alternatively, the **/etc/resolv.conf** file might contain the following entry:

   ```
   nameserver 127.0.0.1
   ```

   The `127.0.0.1` address is the loopback address, which causes the host to access itself as the name server. The **/etc/resolv.conf** file may also contain an entry like the following:

   ```
   domain   domainname
   ```

   In the previous example, the *domainname* value is `austin.century.com`.

5. Perform one of the following steps:

   - Enable the **named** daemon using the **smit stnamed** SMIT fast path. This initializes the daemon with each system startup. Indicate whether you want to start the **named** daemon now, at the next system restart, or both.

   - Edit the **/etc/rc.tcpip** file. Uncomment the line for the **named** daemon by removing the comment (#) symbol from the following line:

     ```
     #start /etc/named ”$src_running”
     ```

     This initializes the daemon with each system startup.

6. If you chose not to initialize the named daemon through SMIT, start the daemon for this session by typing the following command:

   ```
   startsrc –s named
   ```

Configuring a Forward Only Name Server

To configure a forward only name server, use the Web–based System Manager, **wsm**, or use the following procedure, which edits a series of files and then uses SMIT or the command line to start the **named** daemon.

Note:: You can achieve a similar configuration without running a forward only name server. Instead, create an **/etc/resolv.conf** file that contains name server lines that point to the forwarders you wish to use.

1. Edit the **/etc/named.conf** file. If there is no **named.conf** file in the **/etc** directory, copy the **/usr/samples/tcpip/named.conf** sample file into the **/etc** directory and edit it. See the named.conf File Format for TCP/IP in the *AIX 5L Version 5.2 Files Reference* for more information and a detailed example of a **conf** file.

   - Specify the forwarders and forward only lines in the options stanza of the **/etc/named.conf** file listing the IP addresses of the name servers receiving the forwarded requests. For example:

     ```
     options {
          ...
          directory ”/usr/local/domain”;
          forwarders { 192.100.61.1;  129.35.128.222; };
          forward only;
          ...
      };
     ```

   - Specify the loopback zone. For example:

     ```
     zone ”0.0.127.in–addr.arpa” in {
          type master;
          file ”named.abc.local”;
      };
     ```

   - Specify the hint zone. For example:

     ```
     zone ”.” IN {
          type hint;
          file ”named.ca”;
      };
     ```

2. Edit the **/usr/local/domain/named.ca** file. See the DOMAIN Cache File Format for TCP/IP in *AIX 5L Version 5.2 Files Reference* for more information and a detailed example of a cache file. This file contains the addresses of the servers that are authoritative name servers for the root domain of the network. For example:

```
; root name servers.
.          IN  NS      relay.century.com.
 relay.century.com.   3600000    IN    A     129.114.1.2
```

   Note:: All lines in this file must be in Standard Resource Record Format.

3. Edit the **/usr/local/domain/named.abc.local** file. See the DOMAIN Local Data File Format for TCP/IP in the *AIX 5L Version 5.2 Files Reference* for more information and a detailed example of a local data file.

   a. Specify the start of authority (SOA) of the zone and the default time–to–live information. For example:

   ```
   $TTL 3h    ;3 hour

    @ IN SOA venus.abc.aus.century.com. gail.zeus.abc.aus.century.com.  (

                               1       ;serial
                               3600    ;refresh
                               600     ;retry
                               3600000 ;expire
                               86400   ;negative caching TTL
    )
   ```

   b. Specify the name server (NS) record. For example:

   ```
    <tab>      IN    NS     venus.abc.aus.century.com.
   ```

   c. Specify the pointer (PTR) record.

   ```
   1      IN    PTR    localhost.
   ```

   Note:: All lines in this file must be in Standard Resource Record Format.

4. Create an **/etc/resolv.conf** file by typing the following command:

   ```
   touch /etc/resolv.conf
   ```

   The presence of this file indicates that the host should use a name server, not the **/etc/hosts** file, for name resolution.

   Alternatively, the **/etc/resolv.conf** file might contain the following entry:

   ```
   nameserver 127.0.0.1
   ```

   The `127.0.0.1` address is the loopback address, which causes the host to access itself as the name server. The **/etc/resolv.conf** file can also contain an entry such as:

   ```
   domain   domainname
   ```

   In the previous example, the *domainname* value is `austin.century.com`.

5. Perform one of the following steps:

   – Enable the **named** daemon using the **smit stnamed** SMIT fast path. This initializes the daemon with each system startup. Indicate whether you want to start the **named** daemon now, at the next system restart, or both.

   – Edit the **/etc/rc.tcpip** file. Uncomment the line for the **named** daemon by removing the comment (#) symbol from the following line:

   ```
   #start /etc/named ″$src_running″
   ```

   This initializes the daemon with each system startup.

6. If you chose not to initialize the **named** daemon through SMIT, start the daemon for this session by typing the following command:

   ```
   startsrc –s named
   ```

Configuring a Host to Use a Name Server

To configure a host to use a name server, use the Web–based System Manager, **wsm**, or use the following procedure.

1. Create an **/etc/resolv.conf** file by running the following command:

```
touch /etc/resolv.conf
```

2. On the first line of the **/etc/resolv.conf** file, type the word **domain** followed by the full name of the domain that this host is in. For example:

```
domain abc.aus.century.com
```

3. On any blank line below the `domain` line, type the word **nameserver**, followed by at least one space, followed by the dotted decimal Internet address of the name server that this host is to use (the name server must serve the domain indicated by the `domain` statement). You can have up to 16 name server entries. For example, your **/etc/resolv.conf** file might contain the entries:

   ```
   nameserver 192.9.201.1
    nameserver 192.9.201.2
   ```

   The system queries the name servers in the order listed.

   ```
   search domainname_list
   ```

   Alternatively, the search keyword could be used to specify the order in which the resolver will query the domain list. In this case, `domainname_list` values are `abc.aus.century.com` and `aus.century.com`. The `domainname_list` could contain a maximum of six domain names, each separated by a space.

4. Assuming the name server is operational, you can test the communication between the host and the name server by typing the following command:

   ```
   host   hostname
   ```

   Use the name of a host that should be resolved by the name server to see if the process is working. The output you receive should appear similar to the following:

   ```
   brown.abc.aus.century.com is 129.35.145.95
   ```

Other configuration tasks are shown in the following table.

Task	**SMIT fast path**	**Command or file**	Web-based System Manager Management Environment
Create an **/etc/resolv.conf** File	**smit stnamerslv2**	create and edit **/etc/resolv.conf** [1]	
List All the Name Servers Used by a Host	**smit lsnamerslv**	view **/etc/resolv.conf**	Software —> **Network** —> **TCPIP (IPv4 and IPv6)** —> **TCPIP Protocol Configuration** —> **TCP/IP** —> **Configure TCP/IP** —> **Advanced Methods** —> **Hosts File** —> **Contents of /etc/hosts file**.

Add a Name Server	**smit mknamerslv**	edit **/etc/resolv.conf** [2]	Software —> **Network** —> **TCPIP (IPv4 and IPv6)** —> **TCPIP Protocol Configuration** —> **TCP/IP** —> **Configure TCP/IP** —> **Advanced Methods** —> **DNS**. In the **Name Server IP Address** field, type the *IP Address*. Click **Add** —> **OK**.
Remove a Name Server	**smit rmnamerslv**	edit **/etc/resolv.conf**	Software —> **Network** —> **TCPIP (IPv4 and IPv6)** —> **TCPIP Protocol Configuration** —> **TCP/IP** —> **Configure TCP/IP** —> **Advanced Methods** —> **DNS**. Select a name server in **Name server to search**. Click **Delete** —> **OK**.
Start/Restart Using Domain Name Resolution	**smit stnamerslv**		Software —> **Network** —> **TCPIP (IPv4 and IPv6)** —> **TCPIP Protocol Configuration** —> **TCP/IP** —> **Configure TCP/IP** —> **Advanced Methods** —> **DNS**. Select the **Enable domain name resolution using Domain Name Service (DNS)** check box. Click **OK**.
Stop Using Domain Name Resolution	**smit spnamerslv**		Software —> **Network** —> **TCPIP (IPv4 and IPv6)** —> **TCPIP Protocol Configuration** —> **TCP/IP** —> **Configure TCP/IP** —> **Advanced Methods** —> **DNS**. Clear the **Enable domain name resolution using Domain Name Service (DNS)** check box. Click **OK**.

Change/Show the Domain	**smit mkdomain**	edit **/etc/resolv.conf**	Software —> **Network** —> **TCPIP (IPv4 and IPv6)** —> **TCPIP Protocol Configuration** —> **TCP/IP** —> **Configure TCP/IP** —> **Advanced Methods** —> **DNS**. —> **Domain name to search**. Click **Add** —> **OK**.
Remove the Domain	**smit rmdomain**	edit **/etc/resolv.conf**	Software —> **Network** —> **TCPIP (IPv4 and IPv6)** —> **TCPIP Protocol Configuration** —> **TCP/IP** —> **Configure TCP/IP** —> **Advanced Methods** —> **DNS**. Select a domain in the **Domain search list**. Click **Delete** —> **OK**.

Configuring Dynamic Zones on the DNS Name Server

The **named** command allows for dynamic updates. The named database and configuration files need to be configured to allow for client machines to issue updates. A zone can be set to dynamic or static. The default zone is static.

To make a zone dynamic, you must add the keyword **allow–update** to that zone's stanza in the **/etc/named.conf** file. The **allow–update** keyword specifies an Internet address match list that defines hosts allowed to submit updates. See the named.conf File Format for TCP/IP in the *AIX 5L Version 5.2 Files Reference* for more information and a detailed example of a **conf** file. In the following example, all hosts are allowed to update the dynamic zone:

```
zone ”abc.aus.century.com” IN {
     type master;
     file ”named.abc.data”;
     allow–update { any; };
 };
```

After a zone is marked dynamic, three modes of security can be initiated:

Unsecured
Allows anyone at anytime to update any information in the zone. **Attention:** Use of this mode is not recommended. It can lead to data loss, data interception, and user frustration. At the least, an unsecured zone should be limited to updates only from specific Internet addresses.

Controlled
Allows for the creation of new information and the replacement of existing information. This is probably the easiest mode to use for a secure transition environment. This mode also requires that all incoming updates be timestamped and have keyed signatures.

Presecured
Requires all updates to existing information be replaced with similar information. Does not allow for the creation of new information. This mode also requires that all incoming updates be timestamped and have keyed signatures.

A dynamic zone defaults to unsecured mode. To use one of the other modes, type **controlled** or **presecured** after the keyword **update–security** in the zone stanza of the **/etc/named.conf** file. This tells the **named** server the level of security to use with that zone. For example:

```
zone ”abc.aus.century.com” IN {
     type master;
     file ”named.abc.data”;
     allow–update { any; };
     update–security controlled;
 };
```

After a mode is selected, the actual data files must be modified for your level of security. In unsecured mode, the data files are used ”as is.” For controlled or presecured mode, you must generate a set of master server/hostname key pairs for each name in the zone. This is done with the **nsupdate** command using the **–g** option. This command generates the key pair (a private and a public key). These keys are needed to authentically sign for updates. After generating all the keys for your list of zone names, you need to add them to the data file. The KEY format is as follows:

```
 Index          ttl           Class           Type           KeyFlags
   Protocol           Algorithm           KeyData
```

where:

Index Specifies the name used to reference the data in the zone.

ttl Specifies the time–to–live (TTL) for this data. This is an optional field.

Class Specifies the class of the data. This is dependent on the zone, but usually it is IN.

Type Indicates the type of the record. In this case, it is KEY.

KeyFlags Gives named information about the key. 0x0000 defines the typical key record used for a host. 0x0100 defines the key record associated with the zone name.

Protocol Specifies the protocol to use. Currently, there is only one, `0`.

Algorithm Specifies the algorithm of the key. Currently, there is only one, 1. This is the MD5 Private/Public authentication method.

KeyData Indicates the key in base64 representation. The **nsupdate** command generates both the public and private keys in base64 representation. The public key is listed last in the output file.

Example

To ensure security over a host name in a dynamic zone, a line similar to the following needs to be added to the zone file for the zone containing the hostname.

```
bears    4660    IN    KEY    0x0000    0    1    AQOtg......
```

The above example indicates that `bears` has a KEY record defined. Someone wanting to update `bears` would have to sign his update with the private key matching the public key in the database. For the **nsupdate** command to succeed, the private key needs to be placed on the client in a keyfile (defaults to **/etc/keyfile**). It should follow the format:

```
 hostname           mastername           base64           key
```

A similar KEY entry is required in the zone definition section. *A zone key is required for both presecured and controlled modes or the mode is considered to be unsecured*. This can be done as shown in the previous `bears` example, but the private key is left for the administrator to use with the **nsupdate** command's administrative mode.

1. To generate a key pair using the **nsupdate** command, type the following:

   ```
    nsupdate –g –h       ZoneName       –p       ServerName       –k
   AdminKeyFile
   ```

   This generates a key for the zone. In this example, **nsupdate** is linked to **nsupdate4**, which can be done by typing the following:

```
ln –fs /usr/sbin/nsupdate4 /usr/sbin/nsupdate
```

2. Place the last key of the pair in the beginning section for the zone as follows:

   ```
   IN     KEY     0x0100  0  1   Key
   ```

The entry for the **named.abc.data** file is as follows:

```
$TTL 3h    ;3 hour

 @ IN    SOA     venus.abc.aus.century.com.
gail.zeus.abc.aus.century.com. (
                 1       ;serial
                 3600    ;refresh
                 600     ;retry
                 3600000 ;expire
                 86400   ;negative caching TTL
 )
         IN      NS      venus.abc.aus.century.com.
         IN      KEY     0x0100  0 1
AQPlwHmIQeZzRk6Q/nQYhs3xwnhfTgF/8YlBVzKSoKxVKPNLINnYW0mB7attTcfhHaZZcZr4u
/vDNikKnhnZwgn/
 venus   IN      A       192.9.201.1
 earth   IN      A       192.9.201.5
 mars    IN      A       192.9.201.3
```

3. The zone is now ready to be loaded by refreshing the name server. Place the AdminKeyFile on the client or DHCP server that is updating the zone. The zone key contained in the AdminKeyFile can be used to apply updates and maintenance operations to the name server.

BIND 9

BIND 9 offers the following two security measures for **named**:

- Transaction Signatures (TSIG) on page 4-86
- Signature (SIG) on page 4-88

The name server with BIND 9, by default, does not allow dynamic updates to authoritative zones, similarly to that of BIND 8.

Transaction Signatures (TSIG)

BIND 9 primarily supports TSIG for server–to–server communication. This includes zone transfer, notify, and recursive query messages. TSIG is also useful for dynamic updates. A primary server for a dynamic zone should use access control to control updates, but IP–based access control is insufficient.

By using key base encryption rather than the current method of access control lists, TSIG can be used to restrict who can update to the dynamic zones. Unlike the Access Control List (ACL) method of dynamic updates, the TSIG key can be distributed to other updaters without having to modify the configuration files on the name server, which means there is no need for the name server to reread the configuration files.

It is important to note that BIND 9 does not have all the keywords implemented in BIND 8. In this example, we use the simple master configuration from BIND 8.

Note:: To use named 9, you must relink the symbolic link to the **named** daemon to **named9**, and **nsupdate** to **nsupdate9** by running the following commands:

1. `ln –fs /usr/sbin/named9 /usr/sbin/named`
2. `ln –fs /usr/sbin/nsupdate9 /usr/sbin/nsupdate`

1. Generate the key using the **dnssec–keygen** command:

```
dnssec–keygen –a HMAC–MD5 –b 128 –n HOST keyname
```

– `HMAC–MD5` is the algorithm used for encryption

– `128` is the length of the key to use (or number of bits)

– `HOST`: `HOST` is the TSIG keyword used to generate a host key for shared key encryption.

The command

```
dnssec–keygen –a HMAC–MD5 –b 128 –n HOST venus–batman.abc.aus.century.com
```

would produce two key files, as follows:

```
Kvenus–batman.abc.aus.century.com.+157+35215.key
 Kvenus–batman.abc.aus.century.com.+157+35215.private
```

- 157 is the algorithm used (HMAC–MD5)
- 35215 is the finger print, which is useful in DNNSEC because multiple keys per zone are allowed

2. Add the entry to named.conf on the master name server:

```
// TSIG Key
 key venus–batman.abc.aus.century.com. {
         algorithm hmac–md5;
         secret ”+UWSvbpxHWFdNwEAdy1Ktw==”;
 };
```

Assuming HMAC–MD5 is being used, both keyfiles contain the shared key, which are stored as the last entry in the files. Find a secure way to copy the shared secret key to the client. You do not need to copy the keyfile, just the shared secret key.

Following is the entry for file **Kvenus–batman.abc.aus.century.com.+157+35215.private**:

```
Private–key–format: v1.2
 Algorithm: 157 (HMAC_MD5)
 Key: +UWSvbpxHWFdNwEAdy1Ktw==
```

Below is an example of the **named.conf** file for the master name server. The zone `abc.aus.century.com` allows zone transfer and dynamic updates only to servers with the key `venus–batman.abc.aus.century.com`. Do the same to the reverse zone, which requires updaters to have the shared key.

```
// TSIG Key
 key venus–batman.abc.aus.century.com. {
         algorithm hmac–md5;
         secret ”+UWSvbpxHWFdNwEAdy1Ktw==”;
 };

 options {
         directory ”/usr/local/domain”;
 };

 zone ”abc.aus.century.com” in {
         type master;
         file ”named.abc.data”;
         allow–transfer { key venus–batman.abc.aus.century.com.;};
         allow–update{ key venus–batman.abc.aus.century.com.; };
 };
```

Because zones transfers are now restricted to those that have a key, the slave name server’s **named.conf** file must also be edited. All requests to 192.9.201.1(venus.abc.aus.century.com) are signed by a key. Note the name of the key (venus–batman.abc.aus.century.com.) must match those on the servers who use them.

Below is an example of the **named.conf** file on the slave name server:

```
// TSIG Key
 key venus–batman.abc.aus.century.com. {
         algorithm hmac–md5;
         secret ”+UWSvbpxHWFdNwEAdy1Ktw==”;
 };

 server 192.9.201.1{
         keys { venus–batman.abc.aus.century.com.;};
 };

 options {
         directory ”/usr/local/domain”;
 };

 zone ”abc.aus.century.com” IN {
     type slave;
     file ”named.abc.data.bak”;
     masters { 192.9.201.1; };
 };
```

Signature (SIG)

BIND 9 partially supports DNSSEC SIG transaction signatures as specified in RFC 2535. SIG uses public and private keys to authenticate messages.

SIG records allow administers to sign their zone data, thereby stating that it is authentic.

Securing the root zone

Assume that other name servers on the internet are not using BIND 9, and you want to secure your zone data and allow other servers to verify your zone data. You want to state that your zone (in our case `aus.century.com`) is a secure root, and will validate any secure zone data below it.

1. Generate the keys using the **dnssec–keygen** command:

   ```
   dnssec–keygen –a RSA –b 512 –r /usr/sbin/named –n ZONE aus.century.com.
   ```

   Note:: RSA encryption can be used as the algorithm to generate the key if OpenSSL is installed, although you must first relink the DNS library to a secured DNS library by running the following command:

   ```
   ln –fs /usr/lib/libdns_secure.a /usr/lib/libdns.a
   ```

   – ZONE: ZONE is the DNSSEC keyword used to generate zone keys for private/public key encryption

   – The `r` flag specifies a random device

2. Add the public key entry similar to the **named.conf** file. The entry used in our case follows. Below are the contents of key file **Kaus.century.com.+001+03254.key**.

   ```
   abc.aus.century.com. IN KEY 256 3 1
   AQOnfGEAg0xpzSdNRe7KePq3Dl4NqQiq7HkwKl6TygUfaw6vz6ldmauB4UQFcGKOyL68/Zv5Z
   nEvyB1fMTAaDLYz
   ```

   The public key is contained in the file **Kzonename.+algor.+fingerprint.key**, or in our case `Kaus.century.com.+001+03254.key`. You have to remove the class IN and type KEY as well as quote the key. Once you add this entry to the **/etc/named.conf** file and refresh the name server, the zone `aus.century.com` is a secure root.

```
trusted–keys {
         aus.century.com. 256 3 1
”AQOnfGEAg0xpzSdNRe7KePq3Dl4NqQiq7HkwKl6Tyg
 Ufaw6vz6ldmauB 4UQFcGKOyL68/Zv5ZnEvyB1fMTAaDLYz”;
 };
 options {
         directory ”/usr/local/domain”;
 };

 zone ”abc.aus.century.com” in {
         type master;
         file ”named.abc.data.signed”;
         allow–update{192.9.201.1;};
 };
```

Applying the Chain of Trust

Now that you have a secured root, you can secure the rest of your child zones. In this case, we are working to secure the zone `abc.aus.century.com`. Follow these steps to secure your remaining child zones:

1. Generate the key pairs using the **dnssec–keygen** command:

   ```
   dnssec–keygen –a RSA –b 512 –r /usr/sbin/named –n ZONE
   abc.aus.century.com.
   ```

   – The `r` flag specifies a random input file

2. Make a keyset by running the **dnssec–makekeyset** command:

   ```
   dnssec–makekeyset –t 172800   Kabc.aus.century.com.+001+11515.key
    where  Kabc.aus.century.com.+001+03254.key is your own public key.
   ```

   This creates a keyset file called **keyset–abc.aus.century.com**.

3. Send this keyset file to the parent zone to get it signed. In this case, our parent zone is the secure root zone `aus.century.com`.

4. The parent must sign the key using its private key.

   ```
   dnssec–signkey keyset–abc.aus.century.com.
   Kaus.century.com.+001+03254.private
   ```

   This will generate a file called **signedkey–abc.aus.century.com**, and the parent will need to send this file back to the child zone.

5. On the child name server for zone `abc.aus.century.com`, add `$INCLUDE Kabc.aus.century.com.+001+11515.key` to the plain zone file `named.abc.data`. Remember to place the **signedkey–abc.aus.century.com** file in the same location as the zone file **named.abc.data**. When the zone is signed in the following step, the program will know to include **signedkey–abc.aus.century.com**, which was received from the parent.

   ```
   $TTL 3h    ;3 hour

    @ IN    SOA     venus.abc.aus.century.com.
   gail.zeus.abc.aus.century.com.  (
                    1       ;serial
                    3600    ;refresh
                    600     ;retry
                    3600000 ;expire
                    86400   ;negative caching TTL
    )
    $INCLUDE Kabc.aus.century.com.+001+03254.key
   ```

6. Sign the zone using the **dnssec–signzone** command:

   ```
   dnssec–signzone –o abc.aus.century.com. named.abc.data
   ```

7. Modify the **named.conf** file on the child zone `abc.aus.century.com` to use the new signed zone file (`named.abc.data.signed`). For example:

```
options {
         directory ”/usr/local/domain”;
 };

 zone ”abc.aus.century.com” in {
         type master;
         file ”named.abc.data.signed”;
         allow–update{192.9.201.1;};
 };
```

8. Refresh the name server.

For information on troubleshooting, see Name Resolution Problems on page 4-231.

Planning and Configuration for LDAP Name Resolution (SecureWay Directory Schema)

The Lightweight Directory Access Protocol (LDAP) is an open industry standard that defines a method for accessing and updating information in a directory. An LDAP schema defines the rules for ordering data. The **ibm–HostTable** object class, part of the SecureWay Directory schema, can be used to store the name–to–Internet–address mapping information for every host on the network.

The **ibm–HostTable** object class is defined as follows:

```
Object Class name:     ibm–HostTable
 Description:           Host Table entry which has a collection of hostname 
to
                        IP address mappings.
 OID:                   TBD
 RDN:                   ipAddress
 Superior object class: top
 Required Attributes:   host, ipAddress
 Optional Attributes:   ibm–hostAlias, ipAddressType, description
```

The attribute definitions follow:

```
Attribute Name: ipAddress
 Description:    IP Address of the hostname in the Host Table
 OID:            TBD
 Syntax:         caseIgnoreString
 Length:         256
 Single Valued:  Yes
 Attribute Name: ibm–hostAlias
 Description:    Alias of the hostname in the Host Table
 OID:            TBD
 Syntax:         caseIgnoreString
 Length:         256
 Single Valued:  Multi–valued
 Attribute Name: ipAddressType
 Description:    Address Family of the IP Address (1=IPv4, 2=IPv6)
 OID:            TBD
 Syntax:         Integer
 Length:         11
 Single Valued:  Yes
 Attribute Name: host
 Description:    The hostname of a computer system.
 OID:            1.13.18.0.2.4.486
 Syntax:         caseIgnoreString
 Length:         256
 Single Valued:  Multi–valued
 Attribute Name: description
 Description:    Comments that provide a description of a directory object 
entry.
 OID:            2.5.4.13
 Syntax:         caseIgnoreString
 Length:         1024
 Single Valued:  Multi–valued
```

Use the following procedure to configure the LDAP server compliant with the SecureWay Directory schema, for storing the name–to–Internet–address mapping host information.

1. Add a suffix on the LDAP server. The suffix is the starting point of the hosts database. For example, ”cn=hosts”. This can done using the web–based IBM SecureWay Directory Server Administration tool.

2. Create an LDAP Data Interchange Format (LDIF) file. This can be done manually or with the **hosts2ldif** command, which creates a LDIF file from the **/etc/hosts** file. See the **hosts2ldif Command** in the *AIX 5L Version 5.2 Commands Reference* for more information. The following is a sample LDIF file:

```
dn: cn=hosts
 objectclass: top
 objectclass: container
 cn: hosts
 dn: ipAddress=1.1.1.1, cn=hosts
 host: test
 ipAddress: 1.1.1.1
 objectclass: ibm–HostTable
 ipAddressType: 1
 ibm–hostAlias: e–test
 ibm–hostAlias: test.austin.ibm.com
 description: first ethernet interface
 dn: ipAddress=fe80::dead, cn=hosts
 host: test
 ipAddress: fe80::dead
 objectclass: ibm–HostTable
 ipAddressType: 2
 ibm–hostAlias: test–ll
 ibm–hostAlias: test–ll.austin.ibm.com
 description: v6 link level interface
```

3. Import the hosts directory data from the LDIF file on the LDAP server. This can be done with the **ldif2db** command or through the web–based SecureWay Directory Server Administration tool.

To configure the client to access the hosts database on the LDAP server, using the LDAP mechanism, follow the steps below:

1. Create the **/etc/resolv.ldap** file. See the resolv.ldap File Format for TCP/IP in the *AIX 5L Version 5.2 Files Reference* for more information and a detailed example of a **resolv.ldap** file.
2. Change the default name resolution through the **NSORDER** environment variable, the **/etc/netsvc.conf** file, or the **/etc/irs.conf** file. See the netsvc.conf File Format for TCP/IP or the irs.conf File Format for TCP/IP in the *AIX 5L Version 5.2 Files Reference* for more information.

Although still supported, the use of `ldap` mechanism is deprecated. This existing `ldap` mechanism works with SecureWay Directory Schema. AIX 5.2 offers the new naming mechanism, `nis_ldap` (NIS_LDAP), which works with the RFC 2307 schema. Use of the `nis_ldap` mechanism instead of the `ldap` mechanism is recommended. For information on `nis_ldap` name resolution, see Planning and Configuring NIS_LDAP Name Resolution (RFC 2307 schema) on page 4-92 .

Planning and Configuring NIS_LDAP Name Resolution (RFC 2307 schema)

AIX 5.2 offers a new naming mechanism called NIS_LDAP. The difference between the existing LDAP mechanism and the new NIS_LDAP mechanism is in the LDAP schema (the set of attributes and objectclasses that determine how attributes are grouped together for describing an entity). The existing LDAP mechanism works with the SecureWay Directory schema compliant LDAP server and it supports only the host naming service. The NIS_LDAP mechanism works with the RFC 2307 schema compliant LDAP server, and it supports all the NIS services: users and groups, hosts, services, protocols, networks, and netgroup. RFC 2307 defines a set of attributes and objectclasses that can be used to describe network information services, including users and groups.

To configure the LDAP server, you will need to setup the LDAP server and migrate the required data to the server.

1. Use the **mksecldap** command to set up a server. The `nis_ldap` mechanism works only with the RFC 2307 schema. While setting up the LDAP server, the **mksecldap** command should be invoked with either the `–S rfc2307` or `–S rfc2307aix` option (not the `–S aix` option, which specifies the SecureWay Directory schema). By default,

the **mksecldap** command migrates users and groups defined on the local system to the LDAP server. If you want to disable this migration, use the `–u NONE` option.

```
mksecldap –s –a cn=admin –p adminpwd –S rfc2307aix
```

This sets up an LDAP server with administrator DN being `cn=admin` and password being `adminpwd`. The default suffix, `cn=aixdata`, is also added to the **/etc/slapd32.conf** file, the LDAP server configuration file.

By default, the **mksecldap** command migrates users and groups defined on the local system to the LDAP server. If you want to disable this migration, use the `–u NONE` option, which prevents the migration of local users and groups to the LDAP server, so that you can only add NIS users and groups later.

```
mksecldap –s –a cn=admin –p adminpwd –u NONE
```

For more information on the **mksecldap** command, see the command description in *AIX 5L Version 5.2 Commands Reference*.

2. Migrate the NIS data. Use the **nistoldif** command from the NIS server to migrate the NIS maps to the LDAP server. The **nistoldif** command can also be used to migrate data from flat files.

   Run the **nistoldif** command on a system that contains NIS data that needs to be migrated to the LDAP server.

   ```
   nistoldif –h server1.ibm.com –a cn=admin –p adminpwd –d cn=aixdata
   ```

   This migrates the NIS maps from the local system to the LDAP server, `server1.ibm.com`. The NIS data is placed under the `cn=aixdata` DN. You can also run the **nistoldif** command to migrate data from flat files on any system to the LDAP server. The flat files will be used for any maps missing from the NIS server.

   For more information on the **nistoldif** command, see the command description in *AIX 5L Version 5.2 Commands Reference*.

   Note:: Names are represented by the `cn` attribute of the LDAP server. The `cn` attribute defined by RFC 2307 is not case–sensitive. Names that differ only by case will be merged on the server. Matches are also not case–sensitive. Searching for `TCP`, `tcp`, or `Tcp` would all return the protocol entry for TCP.

To configure the LDAP client to access names from the LDAP server, run the **mksecldap** command with client setup options.

1. The **mksecldap** command saves the LDAP servername, port, admindn, password, and basedn to the **/etc/security/ldap/ldap.cfg** file, which is read by the **secldapclntd** daemon at its startup time. The **mksecldap** command starts the **secldapclntd** daemon automatically, if the setup is successful.

   See the **/etc/security/ldap/ldap.cfg** file in the *AIX 5L Version 5.2 Files Reference* and the **secldapclntd** daemon in the *AIX 5L Version 5.2 Commands Reference* for more information.

2. The **mksecldap** command adds `nis_ldap` mechanism to the **/etc/netsvc.conf** file and the **/etc/irs.conf** file so that name resolution can be directed to LDAP. You can also manually set the **NSORDER** environment variable to `nis_ldap` to use the NIS_LDAP name resolution.

   ```
   mksecldap –c –a cn=admin –p adminpwd –h server1.ibm.com
   ```

   This sets up the local system to use the `server1.ibm.com` LDAP server. The LDAP server administrator DN and password must be supplied for this client to authenticate to the server. The **/etc/netsvc.conf** and the **/etc/irs.conf** files are updated so that the naming resolution is resolved through NIS_LDAP.

   See the **/etc/netsvc.conf** file format for TCP/IP or the **/etc/irs.conf** file format for TCP/IP in the *AIX 5L Version 5.2 Files Reference* for more information.

3. Naming resolution for users and groups is not controlled by the **/etc/netsvc.conf** or **/etc/irs.conf** files. Rather it is through the **/etc/security/user** file. To enable a LDAP user to login to an AIX system, set the user's `SYSTEM` and `registry` variables to `LDAP` in the **/etc/security/user** file of that client system. You can run the **chuser** command to do this.

   ```
   chuser –R LDAP SYSTEM=LDAP registry=LDAP foo
   ```

   You can configure your system to allow all LDAP users to login to a system. To do so, edit the **/etc/security/user** file. Add `registry = files` to the root stanza. Then add `SYSTEM = LDAP` and `registry = LDAP` to the default stanza.

   For more information on user authentication, refer to LDAP Exploitation of the Security Subsystem of the *AIX 5L Version 5.2 Security Guide*.

TCP/IP Address and Parameter Assignment – Dynamic Host Configuration Protocol (DHCP)

Transmission Control Protocol/Internet Protocol (TCP/IP) enables communication between machines with configured addresses. Part of the burden a network administrator must face is address assignment and parameter distribution for all machines on the network. Commonly, this is a process in which the administrator dictates the configuration to each user, allowing the user to configure his own machine. However, misconfigurations and misunderstandings can generate service calls that the administrator must deal with individually. The Dynamic Host Configuration Protocol (DHCP) gives the network administrator a method to remove the end user from this configuration problem and maintain the network configuration in a centralized location.

DHCP is an application–layer protocol that allows a client machine on the network, to get an IP address and other configuration parameters from the server. It gets information by exchanging packets between a daemon on the client and another on the server. Most operating systems now provide a DHCP client in their base package.

To obtain an address, the DHCP client daemon (**dhcpcd**) broadcasts a DHCP discover message, which is received by the server and processed. (Multiple servers can be configured on the network for redundancy.) If a free address is available for that client, a DHCP offer message is created, This message contains an IP address and other options that are appropriate for that client. The client receives the server DHCP offer and stores it while waiting for other offers. When the client chooses the best offer, it broadcasts a DHCP request that specifies which server offer it wants.

All configured DHCP servers receive the request. Each checks to see if it is the requested server. If not, the server frees the address assigned to that client. The requested server marks the address as assigned and returns a DHCP acknowledgement, at which time, the transaction is complete. The client has an address for the period of time (lease) designated by the server.

When half of the lease time is used, the client sends the server a *renew* packet to extend the lease time. If the server is willing to renew, it sends a DHCP acknowledgement. If the client does not get a response from the server that owns its current address, it broadcasts a DHCP rebind packet to reach the server if, for example, the server has been moved from one network to another. If the client has not renewed its address after the full lease time, the interface is brought down and the process starts over. This cycle prevents multiple clients on a network from being assigned the same address.

The DHCP server assigns addresses based on keys. Four common keys are network, class, vendor, and client ID. The server uses these keys to get an address and a set of configuration options to return to the client.

network
: Identifies which network segment the packet came from. The network key allows the server to check its address database and assign an address by network segment.

class
: Is completely client configurable. It can specify an address and options. This key can be used to denote machine function in the network or to describe how machines are grouped for administrative purposes. For example, the network administrator might want to create a `netbios` class that contains options for NetBIOS clients or an `accounting` class that represents Accounting department machines that need access to a specific printer.

vendor
: Helps identify the client by its hardware/software platform (for example, a Windows 95 client or an OS/2 Warp client).

client ID
: Identifies the client either through the machine host name or its medium access control (MAC) layer address. The client ID is specified in the configuration file of the **dhcpcd** daemon. Also, the client ID can be used by

the server to pass options to a specific client or prohibit a particular client from receiving any parameters.

These keys can be used by the configuration either singularly or in combinations. If multiple keys are provided by the client and multiple addresses can be assigned, only one is chosen, and the option set is derived from the chosen key first. For more detailed information about the selection of keys and addresses, see Configuring DHCP on page 4-99.

A relay agent is needed so initial broadcasts from the client can leave the local network. This agent is called the BOOTP relay agent. The relay agents act as forwarding agents for DHCP and BOOTP packets.

The DHCP Server

Beginning with AIX 4.3.1, the DHCP server has been segmented into three main pieces, a database, a protocol engine, and a set of service threads, each with its own configuration information.

The DHCP Database

The **db_file.dhcpo** database is used to track clients and addresses and for access control (for example, allowing certain clients on some networks but not others, or disabling BOOTP clients on a particular network). Options are also stored in the database for retrieval and delivery to clients. The database is implemented as a dynamically loadable object, which allows for easy server upgrade and maintenance.

Using the information in the configuration file, the database is primed and verified for consistency. A set of checkpoint files handles updates to the database and reduces the overhead of writes to the main storage file. The database also contains the address and option pools, but these are static and are discussed in Configuring DHCP on page 4-99.

The main storage file and its back up are flat ASCII files that can be edited. The format for the database main storage files are:

```
DF01
 ″   CLIENT ID  ″ ″   0.0.0.0  ″   State       LeaseTimeStart
LeaseTimeDuration       LeaseTimeEnd
   ″   Server IP Address  ″ ″   Class ID  ″ ″   Vendor ID  ″ ″   Hostname
″ ″   Domain Name  ″
 ″   CLIENT ID  ″ ″   0.0.0.0  ″   State       LeaseTimeStart
LeaseTimeDuration       LeaseTimeEnd
   ″   Server IP Address  ″ ″   Class ID  ″ ″   Vendor ID  ″ ″   Host Name
″ ″   Domain Name  ″
...
```

The first line is a version identifier for the file: `DF01c`. The lines that follow are client record definition lines. The server reads from the second line to the end of the file. (The parameters in quotes must be enclosed in quotes.)

`″CLIENT ID ″`
The ID the client uses to represent itself to the server.

`″0.0.0.0 ″` is the IP address currently assigned to the DHCP server. If no address has been assigned, it is”0.0.0.0”.

State The current state of the client. The DHCP protocol engine contains the allowable set, and the states are maintained in the DHCP database. The number next to *State* represents its value. The states can be:

(1) FREE Represents addresses that are available for use. In general, clients do not have this state unless they have no address assigned. **dadmin** and the output from **lssrc** report this state as ”Free”.

(2) BOUND Indicates client and address are tied and that the client has been assigned this address for some amount of time. **dadmin** and the output from **lssrc** report this state as ”Leased”.

(3) EXPIRED Indicates the client and address are tied together, but only for informational purposes, in a similar manner to released addresses. The expired state, however, represents clients that let their leases expire. An expired address is available for use and is reassigned after all free addresses are unavailable and before released addresses are reassigned. **dadmin** and the output from **lssrc** report this state as ”Expired”.

(4) RELEASED Indicates the client and address are tied for informational purposes only. The DHCP protocol suggests that DHCP servers maintain information about the clients it has served for future reference (mainly to try giving the same address to that client that has been assigned that address in the past). This state indicates that the client has released the address. The address is available for use by other clients, if no other addresses are available. **dadmin** and the output from **lssrc** report this as ”Released”.

(5) RESERVED Indicates client and address are tied, but loosely. The client has issued a DHCP discover message and the DHCP server has responded, but the client has not yet responded with a DHCP request for that address. **dadmin** and the output from **lssrc** report this state as ”Reserved”.

(6) BAD Represents an address that is in use in the network but has not been handed out by the DHCP server. This state also represents addresses that clients have rejected. This state does not apply to clients.. **dadmin** and the output from **lssrc** report this state as ”Used” and ”Bad”, respectively.

LeaseTimeStart Is the start of the current lease time (in the number of seconds since January 1, 1970).

LeaseTimeDuration
Represents the duration of the lease (in seconds).

LeaseTimeEnd Uses the same format as *LeaseTimeStart*, but it represents the end of the lease. Some configuration options use different values for the start and end of a lease and these values can be overridden by configuration file options. See DHCP Server File Syntax for db_file Database on page 4-121.

” *Server IP Address* ”
Is the IP address of the DHCP server that owns this record.

” *Class ID* ”
” *Vendor ID* ”
” *Host Name* ”
” *Domain Name* ”
Values that the server uses to determine which options are sent to the server (stored as quoted strings). These parameters increase performance because option lists can be pregenerated for these clients when the DHCP server starts up.

Checkpoint Files

The syntax for the checkpoint files is not specified. If the server crashes or you have to shut down and cannot do a normal closing of the database, the server can process the checkpoint and backup files to reconstruct a valid database. The client that is being written to the checkpoint file when the server crashes is lost. The default files are:

/etc/db_file.cr normal database operation

/etc/db_file.crbk
backups for the database

/etc/db_file.chkpt and **/etc/db_file.chkpt2**
rotating checkpoint files

The DHCP server for AIX 4.3.1 and later is threaded. To maintain high throughput, database operations (including save operations) are thread–efficient. When a save is requested, the existing checkpoint file is rotated to the next checkpoint file, the existing database file is copied to the backup file, and the new save file is created. Each client record is then logged and a bit is toggled to indicate that the client should use the new checkpoint file for logging. When all client records are recorded, the save is closed, and the backup and old checkpoint files are deleted. Clients can still be processed and, depending on whether the client record has been saved, database changes go into a new save file or to a new checkpoint file.

The DHCP Protocol Engine

For AIX 4.3.1 and later, the DHCP protocol engine has been updated to RFC 2131, but is still compatible with RFC 1541. (The server can also process options as defined in RFC 2132.) The protocol engine uses the database to determine what information is returned to the client.

The configuration of the address pools have some configuration options that affect the state of each machine. For example, the DHCP server pings addresses before it hands them out. The amount of time the server waits for a response is now configurable for each address pool.

DHCP Threaded Operations

The last piece of the DHCP server is actually a set of operations that are used to keep things running. Since the DHCP server is threaded, these operations are actually set up as threads that occasionally do things to make sure everything is together.

The first thread, the **main** thread, handles the SRC requests (such as **startsrc**, **stopsrc**, **lssrc**, **traceson**, and **refresh**). This thread also coordinates all operations that affect all threads and handles signals. For example,

- A SIGHUP (–1) causes a refresh of all databases in the configuration file.
- A SIGTERM (–15) will cause the server to gracefully stop.

The next thread, the **dadmin** thread, interfaces with **dadmin** client program and the DHCP server. The **dadmin** tool can be used to get status as well as modify the database to avoid editing the database files manually. Previous versions of the DHCP server prevented any clients from getting addresses if a status request was running. With the addition of the **dadmin** and **src** threads, the server can handle service requests and still handle client requests.

The next thread is the **garbage** thread, which runs timers that periodically clean the database, save the database, purge clients that do not have addresses, and remove reserved addresses that have been in reserve state for too long. All these timers are configurable (see Configuring DHCP on page 4-99). The other threads are packet processors. The number of these is configurable; the default is 10. Each of these can handle a request from a DHCP client. The number of packet processors required is somewhat load– and machine–dependent. If the machine is used for other services than DHCP, it is not wise to start up 500 threads.

Planning DHCP

To use this protocol, the network administrator needs to set up a DHCP server and configure BOOTP relay agents on links that do not have a DHCP server. Advance planning can reduce DHCP load on the network. For example, one server can be configured to handle all your clients, but all packets must be passed through it. If you have a single router between two large networks, it is wiser to place two servers in your network, one on each link.

Another aspect to consider is that DHCP implies a pattern of traffic. For example, if you set your default lease time to fewer than two days and your machines are powered off for the weekend, Monday morning becomes a period of high DHCP traffic. Although DHCP traffic does not cause huge overhead for the network, it needs to be considered when deciding where to place DHCP servers on a network and how many to use.

After enabling DHCP to get the client on the network, a client has no requirement to enter anything. The DHCP client, `dhcpcd`, reads the **dhcpcd.ini** file, which contains information on logging and other parameters needed to start running. After installation, decide which method to use for TCP/IP configuration: minimum configuration or DHCP. If DHCP is selected, choose an interface and specify some optional parameters. To choose the interface, select the keyword **any**, which tells `dhcpcd` to find the first interface that works and use it. This method minimizes the amount of input on the client side.

Configuring DHCP

By default, the DHCP server is configured by reading the **/etc/dhcpsd.cnf** file, which specifies the initial database of options and addresses. The server is started in the **/etc/rc.tcpip** file. It can also be started from Web–based System Manager, from SMIT, or through SRC commands. The DHCP client can be configured by running Web–based System Manager, the System Management Interface Tool (SMIT), or editing a flat ASCII file.

Configuring the DHCP server is usually the hardest part of using DHCP in your network. First, decide what networks you want to have DHCP clients on. Each subnet in your network represents a pool of addresses that the DHCP server must add to its database. For example:

```
database db_file
 {

     subnet 9.3.149.0 255.255.255.0
       {  option 3 9.3.149.1 # The default gateway clients on this network
should use
          option 6 9.3.149.2 # The nameserver clients on this network
should use
       }
     ... options or other containers added later
 }
```

The example above shows a subnet, `9.3.149.0`, with a subnet mask `255.255.255.0`. All addresses in this subnet, 9.3.149.1 through 9.3.149.254, are in the pool. Optionally, a range can be specified on the end of the line or a range or exclude statement can be included in the subnet container. See DHCP Server File Known Options on page 4-107 for common configuration methods and definitions.

The database clause with `db_file` indicates which database method to use for processing this part of the configuration file. Comments begin with a # (pound sign). Text from the initial #, to the end of the line, is ignored by the DHCP server. Each `option` line is used by the server to tell the client what to do. DHCP Server File Known Options on page 4-107 describes the currently supported and known options. See DHCP Server File Syntax for General Server Operation on page 4-116 for ways to specify options that the server does not know about.

If the server does not understand how to parse an option, it uses default methods to send the option to the client. This also allows the DHCP server to send site–specific options that are not RFC defined, but may be used be certain clients or client configurations.

The Configuration File

The configuration file has an address section and an option definition section. These sections use containers to hold options, modifiers, and, potentially, other containers.

A *container* (basically, a method to group options) uses an identifier to classify clients into groups. The container types are subnet, class, vendor, and client. Currently, there is not a generic user–definable container. The identifier uniquely defines the client so that the client can be tracked if, for example, it moves between subnets. More than one container type can be used to define client access.

Options are identifiers that are returned to the client, such as default gateway and DNS address.

Modifiers are single statements that modify some aspect of a container, such as lease time default.

Containers

When the DHCP server receives a request, the packet is parsed and identifying keys determine which containers, options, and addresses are extracted.

The previous example shows a subnet container. Its identifying key is the position of the client in the network. If the client is from that network, then it falls into that container.

Each type of container uses a different option to identify a client:

- The subnet container uses the **giaddr** field or the interface address of the receiving interface to determine from which subnet the client came.
- The class container uses the value in option 77 (User Site Class Identifier).
- The vendor uses the value in option 60 (Vendor Class Identifier).
- The client container uses the option 61 (Client Identifier) for DHCP clients and the **chaddr** field in the BOOTP packet for BOOTP clients.

Except for subnets, each container allows the specification of the value that matchs it, including regular expression matching.

There is also an implicit container, the *global* container. Options and modifiers are placed in the global container unless overridden or denied. Most containers can be placed inside other containers implying a scope of visibility. Containers may or may not have address ranges associated with them. Subnets, by their nature, have ranges associated with them.

The basic rules for containers and subcontainers are:

- All containers are valid at the global level.
- Subnets can not be placed inside other containers.
- Restricted containers cannot have regular containers of the same type within them. (For example, a container with an option that only allows a class of `Accounting` cannot include a container with an option that allows all classes that start with the letter ”a”. This is illegal.)
- Restricted client containers cannot have subcontainers.

Given the above rules, you can generate a hierarchy of containers that segment your options into groups for specific clients or sets of clients.

If a client matches multiple containers, how are options and addresses handed out? The DHCP server receives messages, it passes the request to the database (**db_file** in this case), and a container list is generated. The list is presented in order of depth and priority. Priority is defined as an implicit hierarchy in the containers. Strict containers are higher priority than regular containers. Clients, classes, vendors, and finally subnets are sorted, in that order, and within container type by depth. This generates a list ordered by most specific to least specific. For example:

```
Subnet 1
 ––Class 1
 ––Client 1
 Subnet 2
 ––Class 1
 ––––Vendor 1
 ––––Client 1
 ––Client 1
```

The example shows two subnets, `Subnet 1` and `Subnet 2`. There is one class name, `Class 1`, one vendor name, `Vendor 1`, and one client name, `Client 1`. `Class 1` and `Client 1` are defined in multiple places. Because they are in different containers, their names can be the same but values inside them can be different. If `Client 1` sends a message to the DHCP server from `Subnet 1` with `Class 1` specified in its option list, the DHCP server would generate the following container path:

`Subnet 1, Class 1, Client 1`

The most specific container is listed last. To get an address, the list is examined in reverse hierarchy to find the first available address. Then, the list is examined in forward hierarchy to get the options. Options override previous values unless an option *deny* is present in the container. Also, because `Class 1` and `Client 1` are in `Subnet 1`, they are ordered according to the container priority. If the same client is in `Subnet 2` and sends the same message, the container list generated is:

`Subnet 2, Class 1, Client 1` (at the `Subnet 2` level), `Client 1` (at the `Class 1` level)

`Subnet 2` is listed first, then `Class 1`, then the `Client 1` at the `Subnet 2` level (because this client statement is only one level down in the hierarchy). The hierarchy implies that a client matching the first client statement is less specific than the client matching `Client 1` of `Class 1` within `Subnet 2`.

Priority selected by depth within the hierarchy is not superseded by the priority of the containers themselves. For example, if the same client issues the same message and specifies a vendor identifier, the container list is:

`Subnet 2, Class 1, Vendor 1, Client 1` (at `Subnet 2` level), `Client 1` (at `Class 1` level)

Container priority improves search performance because it follows a general concept that client containers are the most specific way to define one or more clients. The class container holds less specific addresses than a client container; vendor is even less specific; and subnet is the least specific.

Addresses and Address Ranges

Any container type can have associated addresses ranges; subnets must have associated address ranges. Each range within a container must be a subset of the range and must not overlap with ranges of other containers. For example, if a class is defined within a subnet and the class has a range, the range must be a subset of the subnet range. Also, the range within that class container cannot overlap with any other ranges at its level.

Ranges can be expressed on the container line and modified by range and exclude statements to allow for disjoint address sets associated with a container. If you have the top ten addresses and the second ten addresses of a subnet available, the subnet can specify these addresses by range in the subnet clause to reduce both memory use and the chance of address collision with other clients not in the specified ranges.

Once an address has been selected, any subsequent container in the list that contains address ranges is removed from the list along with its children. Network–specific options in removed containers are not valid if an address is not used from within that container.

Options

After the list has been culled to determine addresses, a set of options is generated for the client. In this selection process, options overwrite previously selected options unless a *deny* is encountered, in which case, the denied option is removed from the list being sent to the client. This method allows inheritance from parent containers to reduce the amount of data that must be specified.

Modifiers

Modifiers are items that change some aspect of a particular container, such as access or lease time. Define the address and option pools before modifying the container. The most common modifyers are **leasetimedefault**, **supportBootp**, and **supportUnlistedclients**.

leasetimedefault
Defines the amount of time an address is to be leased to a client.

supportBootp Defines whether or not the server responds to BOOTP clients.

supportUnlistedclients
Indicates whether clients are to be explicitly defined by a client statement to receive addresses. The value for supportUnlistedClients can be **none**, **dhcp**, **bootp**, or **both**. This allows for you to restrict access to bootp client and allow all DHCP clients to get addresses.

Other modifiers are listed in DHCP Server File Syntax for db_file Database.

Logging

After selecting modifiers, the next item to set up is logging. Logging parameters are specified in a container like the database, but the container keyword is **logging_info**. When learning to configure DHCP, it is advisable to turn logging to its highest level. Also, it is best to specify the logging configuration before any other configuration file data to ensure that configuration errors are logged after the logging subsystem is initialized. Use the **logitem** keyword to turn on a logging level or remove the **logitem** keyword to disable a logging level. Other keywords for logging allow the specification of the log filename, file size, and the number of rotating log files.

Server–specific Options

The last set of parameters to specify are server–specific options that allow the user to control the number of packet processors, how often the garbage collection threads are run, and so on..

For example, two server–specific options are:

reservedTime Indicates how long an address stays in the reserved state after sending an OFFER to the DHCP client

reservedTimeInterval
Indicates how often the DHCP server scans through the addresses to see if there are any that have been in the reserved state longer than **reservedTime**.

These options are useful if you have several clients that broadcast DISCOVER messages and, either they do not broadcast their REQUEST message, or their REQUEST message gets lost in the network. Using these parameters keeps addresses from being reserved indefinitely for a noncompliant client.

Another particularly useful option is **SaveInterval**, which indicates how often saves occur. All server–specific options are listed in DHCP Server File Syntax for General Server Operation with the logging keywords.

Performance Considerations

It is important to understand that certain configuration keywords and the structure of the configuration file have an effect on the memory use and performance of the DHCP server.

First, excessive memory use can be avoided by understanding the inheritance model of options from parent to child containers. In an environment that supports no unlisted clients, the administrator must explicitly list each client in the file. When options are listed for any specific client, the server uses more memory storing that configuration tree than when options are inherited from a parent container (for example, the subnet, network, or global containers). Therefore, the administrator should verify whether any options are repeated at the client level within the configuration file and determine whether these options can be specified in the parent container and shared by the set of clients as a whole.

Also, when using the **logItem** entries INFO and TRACE, numerous messages are logged during the processing of every DHCP client message. Appending a line to the log file can be an expensive operation; therefore, limiting the amount of logging improves the performance of the DHCP server. When an error with the DHCP server is suspected, logging can be dynamically re–enabled using either the SRC **traceson** or **dadmin** commands.

Finally, selecting a **numprocessors** value depends on the size of the DHCP–supported network, the **pingTime db_file** configuration parameter, and the typical propagation delay on the network. Because each packet processor thread issues an ICMP Echo Request to verify the status of a server–owned address before offering it to a client, the amount of time that any Echo Response is waited for directly affects the amount of processing time for a DISCOVER message. Essentially, the packet processor thread is able to do nothing more than wait for any response or for the **pingTime** timeout. Lowering the **numprocessors** value improves the response time of the server by lowering the number of client retransmissions, yet still maintaining the ping benefit of the server design.

For best performance, select a **pingTime** based on the propagation delay of any remote networks supported by the DHCP server. Also, select the **numprocessors** value based on this **pingTime** value and the size of the network. Selecting a value that is too small can cause all packet processing threads to be stopped. The server is then caused to wait for any Echo Responses while incoming DHCP client messages are queueing on the server port. This causes the server to handle client messages in batches rather than in a constant stream.

A selected value that is too small can cause all packet processing threads to be stopped waiting for any Echo Responses , which would result in the .

To prevent this situation, set the value for **numprocessors** to a number higher than the estimated number of DISCOVER messages that can be received within one **pingTime** interval during a period of high DHCP client activity. However, do not set the **numprocessors** value so high that it could burden the kernel with thread management.

For example, the values **numprocessors 5** and **pingTime 300** cause poor performance in an environment with a potential 10 DISCOVER messages per second because at peak demand, only 5 messages are handled every 3 seconds. Configure this environment with values similar to **numprocessors 20** and **pingTime 80**.

Customizing a Configuration File

Many networks include multiple client types; for example, a single network may include computers running a variety of operating systems, such as Windows, OS/2, Java OS, and UNIX. Each of these require unique vendor identifiers (the field used to identify the type of machine to the DHCP server). Java OS clients and Thin Client machines can require unique parameters such as bootfiles, and configuration options that need to be tailored specifically for them. Windows 95 computers do not handle Java–specific options well.

Machine–specific options can be encapsulated within vendor containers if the rimary use for certain machines is based on the type of user for those machines. For instance, the development staff might use this operating system's clients for programming, the marketing staff might use the OS/2 clients, sales might use Java OS clients and Thin Client machines, and accounting might use Windows 95 machines. Each of these user families might need different configuration options (different printers, nameservers, or default web servers, and so forth). In this case, such options could be included in the vendor container, since each group uses a different machine type.

If the same machine type is used by multiple groups, placing the options within a subordinate class identifier instead, would allow your marketing managers, for example, to use a specific set of printers that other employees could not access.

Note:: The following fictional example represents part of a configuration file. Comments are preceded by a pound sign (#) and describe how each line defines the installation.

```
vendor ”AIX_CLIENT”
 {
 # No specific options, handles things based on class
 }

 vendor ”OS/2 Client”
 {
 # No specific options, handles things based on class
 }

 vendor ”Windows 95”
 { option 44 9.3.150.3          # Default NetBIOS Nameserver
 }

 vendor ”Java OS”
 { bootstrapserver 9.3.150.4    # Default TFTP server for the Java OS boxes
   option 67 ”javaos.bin”       # The bootfile of the Java OS box
 }

 vendor ”IBM Thin Client”
 { bootstrapserver 9.3.150.5    # Default TFTP server for Thin Client boxes
   option 67 ”thinos.bin”       # Default bootfile for the Thin Client
boxes
 }

 subnet 9.3.149.0 255.255.255.0
 { option 3 9.3.149.1           # The default gateway for the subnet
   option 6 9.3.150.2           # This is the nameserver for the subnet
   class accounting 9.3.149.5–9.3.149.20
   {       # The accounting class is limited to address range
9.3.149.5–9.3.149.20
           # The printer for this group is also in this range, so it is
excluded.
      exclude 9.3.149.15
      option 9 9.3.149.15       # The LPR server (print server)
      vendor ”Windows 95”
      {
      option 9 deny              # This installation of Windows 95 does not
support
                                 # this printer, so the option is denied.
      }
   }
  . . .
 }
```

DHCP and the Dynamic Domain Name System (DDNS)

The DHCP server provides options that enable operation in a DDNS environment. To use DHCP in a DDNS environment, you must set and use a Dynamic Zone on a DNS server.

After the DDNS server is configured, decide if the DHCP server is going to do A–record updates, PTR–record updates, updates for both record types, or none at all. This decision depends on whether a client machine can do part or all of this work.

- If the client can share update responsibility, configure the server to do the PTR–record updates and configure the client to do the A–record updates.
- If the client can do both updates, configure the server to do none.
- If the client cannot do updates, configure the server to do both.

The DHCP server has a set of configuration keywords that allow you to specify a command to run when an update is required. These are:

updatedns (Deprecated.) Represents the command to issue to do any type of update. It iscalled for both the PTR–record and the A–record update.

updatednsA Specifies the command to update the A–record.

updatednsP Specifies the command to update the PTR–record.

These keywords specify executable strings that the DHCP server runs when an update is required. The keyword strings must contain four `%s` (percent symbol, letter s). The first `%s` is the hostname; the second is the domain name; the third is the IP address; and the fourth is the lease time. These are used as the first four parameters for the **dhcpaction** command. The remaining two parameters for the **dhcpaction** command indicate the record to update (A, PTR, NONE, or BOTH) and whether NIM should be updated (NIM or NONIM). See DHCP and Network Installation Management (NIM) Suggestions for more information about NIM and DHCP interaction. For example:

```
updatednsA ″/usr/sbin/dhcpaction ′%s′ ′%s′ ′%s′ ′%s′ A NONIM″
                     # This does the dhcpaction command only on the A
record
 updatednsP ″/usr/sbin/dhcpaction ′%s′ ′%s′ ′%s′ ′%s′ PTR NONIM″
                     # This does the command only on the PTR record
 updatedns ″/usr/sbin/dhcpaction ′%s′ ′%s′ ′%s′ ′%s′ BOTH NIM″
                     # This does the command on both records and updates
NIM
```

The DHCP server also has a set of keywords to remove the DNS entries when a lease is released or expires. The keywords are:

releasednsA Removes the A–record.

releasednsP Removes the PTR–record.

removedns Removes both record types.

These keywords specify executable strings that the DHCP server runs when an address is released or expired. The **dhcpremove** command works similarly to **dhcpaction**, but only takes three parameters:

1. The IP address, specified as a `%s` in the command string
2. Which record to remove (A, PTR, NONE, or BOTH).
3. Whether NIM should be updated (NIM or NONIM).

For example:

```
releasednsA ″/usr/sbin/dhcpremove ′%s′ A NONIM″
                      # This does the dhcpremove command only the A record
 releasednsP ″/usr/sbin/dhcpremove ′%s′ PTR NONIM″
                      # This does the command only on the PTR record
 removedns ″/usr/sbin/dhcpremove ′%s′ BOTH NIM″
                      # This does the command on both records and updates
NIM
```

The **dhcpaction** and **dhcpremove** scripts do some parameter checking, then set up a call to **nsupdate**, which has been updated to work with this operating system’s servers and with OS/2 DDNS servers. See the **nsupdate** command description for more information.

If NIM interaction is **NOT** required by the name update, the DHCP server can be configured to use a socket transfer between the **DHCP** daemon and the **nsupdate** command to improve performance and enable DNS updates to be retried upon failure. To configure this option, the **updateDNSA**, **updateDNSP**, **releaseDNSA**, or the **releaseDNSP** keyword must specify ”nsupdate_daemon” as the first quoted word. The parameters and flags for this update are identical to those that are accepted by the **nsupdate** command. Additionally, the following variable names can be used for substitution:

$hostname
Replaced by the host name of the client on DNS update or the host name previously associated with the client for DNS removal.

$domain
Replaced by the DNS domain for the update or the previously used domain of the client host name for a DNS removal.

$ipadress
Replaced by the IP address to be associated or disassociated from the DHCP client name.

$leasetime
Replaced by the lease time (in seconds).

$clientid
Replaced by the string representation of the DHCP client identifier or the combination hardware type and hardware address for BOOTP clients.

For example:

```
updateDNSA ”nsupdate_daemon –p 9.3.149.2 –h $hostname –d $domain
  –s”d;a;*;a;a;$ipaddress;s;$leasetime;3110400””

 updateDNSP ”nsupdate_daemon –p 9.3.149.2 –r $ipaddress
  –s”d;ptr;*;a;ptr;$hostname.$domain.;s;$leasetime;3110400””

 releaseDNSA ”nsupdate_daemon –p 9.3.149.2 –h $hostname –d $domain
–s”d;a;*;s;1;3110400””

 releaseDNSP ”nsupdate_daemon –p 9.3.149.2 –r $ipaddress
–s”d;ptr;*;s;1;3110400””
```

See the **nsupdate** command description for more information.

Also, administrator–defined policies have been added for hostname exchanges between the server and the clients. By default, the hostname that is returned to the client and used for a DDNS update is option 12 (defined in the server configuration file). Alternatively, the default hostname can be the client–suggested hostname, either through option 81 (the DHCPDDNS option) or through option 12 (the HOSTNAME option). However, the administrator can override the default hostname by using the **hostnamepolicy**, **proxyarec**, and **appenddomain** configuration keywords. These options and their parameters are defined in DHCP Server File Syntax for db_file Database.

DHCP Compatibility with Older Versions

The DHCP server for AIX 4.3.1 and later recognizes the previous versions configuration and database files, **dhcps.ar** and **dhcps.cr**. It parses the old configuration files and generates new database files in the old locations. The old databases are converted automatically to the new file. The configuration file itself is not converted.

The DHCP server database module, **db_file**, can read the old format. The DHCP server can recognize when a database container is not in the configuration file and treats the whole file as configuring the server parameters, logging parameters, and the **db_file** database parameters. Notes:

1. Some old configuration file syntax is deprecated, but is still supported. Other deprecations are as follows:
2. The network container is completely deprecated. To specify correctly, either convert the network clause with a range into a valid subnet container with a subnet address, subnet netmask, and the range. If the network container has subnet containers, remove the network container keyword and its braces and then place the subnet mask in the appropriate place on the line. To start using the database container, group everything that pertains to networks and client access into one database container of type **db_file**.
3. The **updatedns** and **removedns** keywords are deprecated and replaced in favor of specifying the action for A and PTR records separately.
4. The **clientrecorddb** and **addressrecorddb** keywords have been deprecated to **clientrecorddb** and **backupfile**, respectively.
5. The **option sa** and **option ga** keywords have been replaced by **bootstrapserver** and **giaddrfield** keywords, respectively. See DHCP Server File Syntax for General Server Operation on page 4-116 and DHCP Server File Syntax for db_file Database on page 4-121 for more information.

DHCP Server File Known Options

Note:: The options that are shown in the following table as not allowed to be specified (No in the Can Specify? column) can be specified in the configuration file, but are overwritten by the correct value. For a better definition of each option, see RFC 2132.

Option Number	Default Data Type	Can Specify?	Description/Use
0	None	No	The server pads the option field, if necessary.
1	Dotted quad	No	The net mask of the subnet from which the address was drawn.
2	32–bit integer	Yes	Specifies the offset of the client subnet, in seconds from Coordinated Universal Time (UTC).
3	One or more dotted quads	Yes	A list of the default gateways' IP addresses.
4	One or more dotted quads	Yes	A list of time server IP addresses.

5	One or more dotted quads	Yes	A list of name server IP addresses.
6	One or more dotted quads	Yes	A list of DNS IP addresses.
7	One or more dotted quads	Yes	A list of log server IP addresses.
8	One or more dotted quads	Yes	A list of cookie server IP addresses.
9	One or more dotted quads	Yes	A list of LPR server IP addresses.
10	One or more dotted quads	Yes	A list of Impress server IP addresses.
11	One or more dotted quads	Yes	A list of Resource Location server IP addresses.
12	An ASCII string	Yes	A hostname for the client to use.
13	16–bit unsigned integer	Yes	The size of the bootfile.
14	An ASCII string	Yes	The path for Merit Dump file.
15	An ASCII string	Yes	The default DNS domain name.
16	An IP address	Yes	The address of the Swap server.
17	An ASCII string	Yes	The default root path.
18	An ASCII string	Yes	The path to extensions for the client.
19	Yes, No, True, False, 1, 0	Yes	Specify whether IP Forwarding should be turned on.
20	Yes, No, True, False, 1, 0	Yes	Specify whether non–local source routing should be used.
21	One or more pairs of dotted quads, in the form *DottedQuad*: *DottedQuad*	Yes	The filter policies for IP addresses.
22	16–bit unsigned integer	Yes	The maximum size to allow for datagram fragments.
23	8–bit unsigned integer	Yes	The IP time–to–live (TTL).
24	32–bit unsigned integer	Yes	The number of seconds to use in the Path MTU aging timeout.

25	List of one or more 16–bit unsigned integers	Yes	The path MTU Plateau table. Specifies a set of values that represent the MTU sizes to use when using Path MTU discovery.
26	16–bit unsigned integer	Yes	Specifies MTU size for the receiving interface.
27	Yes, No, True, False, 1, 0	Yes	Specifis whether all subnets are local.
28	An IP address (dotted quad)	Yes	Specifies broadcast address for the interface.
29	Yes, No, True, False, 1, 0	Yes	Specifies whether ICMP netmask discovery should be used.
30	Yes, No, True, False, 1, 0	Yes	Specifies whether client should become an ICMP netmask supplier.
31	Yes, No, True, False, 1, 0	Yes	Specifies whether ICMP Router Discovery messages should be used.
32	IP address (dotted quad)	Yes	Specifies address to use for router solicitation.
33	One or more IP address pairs, in the form *DottedQuad*: *DottedQuad*	Yes	Each address pair represents a static route.
34	Yes/No, True/False, 1/0	Yes	Specifies whether trailer encapsulation should be used.
35	32–bit unsigned integer	Yes	ARP cache timeout value.
36	Yes/No, True/False, 1/0	Yes	Spcifies whether Ethernet encapsulation should be used.
37	8–bit unsigned integer	Yes	The TCP time–to–live (TTL).
38	32–bit unsigned integer	Yes	The TCP keep alive interval.
39	Yes/No, True/False, 1/0	Yes	Specifies whether TCP keep alive should be used.
40	An ASCII string	Yes	The NIS default domain.

41	One or more dotted quads	Yes	Specifies the IP addresses of the NIS servers.
42	One or more dotted quads	Yes	Specifies the IP addresses of the NTP servers.
43	hex string of digits, in the form of hex ” *digits* ”, hex ” *digits* ”, or 0x *digits*	Yes, but really only specified with vendor container	Encapsulated option container for the vendor container.
44	One or more dotted quads	Yes	Specifies NetBIOS name server IP addresses.
45	One or more dotted quads	Yes	Specifies NetBIOS datagram distribution server IP addresses.
46	8–bit unsigned integer	Yes	Specifies NetBIOS Node Type.
47	hex string of digits, in form of hex ” *digits* ”, hex ” *digits* ”, or 0x *digits*	Yes	NetBIOS Scope.
48	One or more dotted quads	Yes	Specifies X Windows font server IP addresses.
49	One or more dotted quads	Yes	Specifies X Windows Display Manager.
50	None	No	Requested IP Address, used by client to indicate the address it wants.
51	32–bit unsigned integer	Yes	Lease time for the returned address. By default, the DHCP server uses the **leasetimedefault** keyword, but direct specification of option `51` overrides it.
52	None	No	Option overload. Client uses this to indicate the **sname** and **file** fields of the BOOTP packet may have options.
53	None	No	DHCP server or client uses this option to indicate the type of DHCP message.

54	None	No	DHCP server or client uses this option to indicate the server's address or the server to which the message is directed.
55	None	No	DHCP client uses this to indicate desired options.
56	An ASCII string	Yes	A string the DHCP server sends to the client. In general, this can be used by the DHCP server and client to indicate problems.
57	No	No	DHCP client uses this option to tell the DHCP server the maximum DHCP packet size the client can receive.
58	32–bit unsigned integer	Yes	Specifies the number of seconds until the client should send a renew packet.
59	32–bit unsigned integer	Yes	Specifies the number of seconds until the client should send a rebind packet.
60	None	No	DHCP client uses this option to indicate its vendor type. The DHCP server uses this field to match vendor containers.
61	None	No	DHCP client uses this to uniquely identify itself. The DHCP server uses this field to match client containers.
64	An ASCII string	Yes	Specifies the NIS+ domain.
65	One or more dotted quads	Yes	IP Addresses of NIS+ servers.
66	An ASCII string	Yes	Specifies the TFTP server name. This is a hostname and is used instead of the **siaddr** field if the client understands this option.

67	An ASCII string	Yes	Specifies the bootfile name. This can be used instead of the **bootfile** keyword, which places the file in the **filename** field of the packet.
68	One or more dotted quads, or NONE	Yes	Specifies addresses of home agents.
69	One or more dotted quads	Yes	Specifies default SMTP servers to use.
70	One or more dotted quads	Yes	Specifies default POP3 servers to use.
71	One or more dotted quads	Yes	Specifies default NNTP servers to use.
72	One or more dotted quads	Yes	Specifies default WWW servers to use.
73	One or more dotted quads	Yes	Specifies default Finger servers to use.
74	One or more dotted quads	Yes	Specifies default IRC servers to use.
75	One or more dotted quads	Yes	Specifies default Street Talk servers to use.
76	One or more dotted quads	Yes	Specifies default Street Talk directory assistance servers to use.
77	An ASCII string	Yes	The user site class identifier. The DHCP server uses this field to match class containers.
78	Mandatory Byte, one or more dotted quads	Yes	The SLP directory Agent Option specifies a list of IP addresses for Directory Agents
79	Mandatory Byte, and ASCII string	Yes	The ASCII string is a scope list, which is a comma delimited list, that indicates the scopes that an SLP Agent is configured to use

81	An ASCII string plus other items	No	The DHCP client uses this option to define the policy the DHCP server should use with respect to DDNS.
85	One or more dotted quads	Yes	NDS server option specifies one or more NDS servers for the client to contact for access to the DNS database. Servers should be listed in order of preference.
86	An ASCII string	Yes	NDS tree name option specifies the name of the NDS tree the client will be contacting.
87	An ASCII string	Yes	NDS context option specifies the initial NDS context the client should use.
93	None	No	The DHCP client uses this option to define the client system architecture.
94	None	No	The DHCP client uses this option to define the client network interface identifier.

117	One or more 16–bit unsigned integers	Yes	Name Service Search Option gives the preferred order of integer option code for name services. For example:
			`Name Services value`
			`Domain Name Server Option  6`
			`NIS Option 41`
			`NIS+ Option 65`

118	One dotted quads	No	Subnet Selection Option is an option sent by the client asking the dhcp server to allocate the IP address from the specified subnet.
255	None	No	DHCP server and client use this option to indicate the end of an option list.

Preboot Execution Environment (PXE) Vendor Container Suboption

When supporting a preboot execution environment (PXE) client, the DHCP server passes the following option to the BINLD server, which is used by BINLD to configure itself:

Opt Num	Default Data Type	Can Specify?	Description
7	one dotted quad	Yes	Multicast IP address. Boot server discovery multicast IP address.

The following example shows how this option can be used:

```
pxeservertype    proxy_on_dhcp_server

 Vendor pxeserver
 {
      option   7    9.3.4.68
 }
```

In the above example, the DHCP server informs the client that the proxy server is running on the same machine but is listening on port 4011 for client requests. The vendor container is required here because the BINLD server broadcasts an INFORM/REQUEST message on port 67 with option 60 set to ”PXEServer.” In response, the DHCP server sends the Multicast IP address on which the BINLD has to listen for PXEClient’s request.

Configuration File Examples Supporting PXE Clients

In the following example, the **dhcpsd** server either gives the bootfile name to the PXEClient or it directs the PXEClient to the BINLD server by sending suboptions. The **pxeboofile** keyword is used to create a list of boot files for a given client architecture and major and minor versions of the client system.

```
pxeservertype     dhcp_pxe_binld

 subnet default
 {
      vendor pxe
      {
          option 6 2    # Disable Multicast
          option 8 5 4 10.10.10.1 12.1.1.15 12.5.5.5 12.6.6.6\
                   2 2 10.1.1.10 9.3.4.5 1 1 10.5.5.9\
                   1 1 9.3.149.15\
                   4 0
          option 9 5 ”WorkSpace On Demand” 2 ”Intel”\
                   1 ”Microsoft WindowsNT” 4 ”NEC ESMPRO”
          option 10 2 ”Press F8 to View Menu”
      }
      vendor pxeserver
      {
          option 7 239.0.0.239
      }

 }

 subnet  9.3.149.0  255.255.255.0
 {
      option 3   9.3.149.1
      option 6   9.3.149.15

      vendor pxe
      {
        option    6    4    # bootfile is present in the offer packet
        pxebootfile   1   2   1   os2.one
        pxebootfile   2   2   1   aix.one
      }
 }
```

Each option line in pxe container is used by the server to tell the client what to do. PXE Vendor Container Suboptions describes the currently supported and known PXE sub–options.

DHCP Server File Syntax for General Server Operation

Note:: Time Units (*time_units*) shown in the following table are optional and represent a modifier to the actual time. The default time unit is minutes. Valid values are seconds (1), minutes (60), hours (3600), days (86400), weeks (604800), months (2392000), and years (31536000). The number shown in parentheses is a multiplier applied to the specified value *n* to express the value in seconds.

Keyword	**Form**	**Subcontainers?**	**Default Value**	**Meaning**
database	database *db_type*	Yes	None	The primary container that holds the definitions for the address pools, options, and client access statements. *db_type* is the name of a module that is loaded to process this part of the file. The only value currently available is **db_file**.
logging_info	logging_info	Yes	None	The primary logging container that defines the logging parameters.
logitem	logitem NONE logitem SYSERR logitem OBJERR logitem PROTOCOL logitem PROTERR logitem WARN logitem WARNING logitem CONFIG logitem EVENT logitem PARSEERR logitem ACTION logitem ACNTING logitem STAT logitem TRACE logitem RTRACE logitem START	No	All default to not enabled.	Enables the logging level. Multiple lines are allowed.

numLogFiles	numLogFiles *n*	No	0	Specifies the number of log files to create. The log rotates when the first one fills. *n* is the number of files to create.
logFileSize	logFileSize *n*	No	0	Specifies the size of each log file in 1024–byte units.
logFileName	logFileName *path*	No	None	Specifies the path to the first log file. The original log file is named *filename* or *filename.extension*. The *filename* must be eight or fewer characters. When a file is rotated, it is renamed beginning with the base *filename*, then either appending a number or replacing the extension with a number. For example, if the original file name is `file`, the rotated file name becomes `file01`. If the original file name is `file.log`, it becomes `file.01`.
CharFlag	charflag yes	No	true	Not applicable to this operating system's DHCP server, but the OS/2 DHCP server uses it to produce debug windows.
	charflag true			
	charflag false			
	charflag no			
StatisticSnapShot	StatisticSnapShot *n*	No	–1, never	Specifies how often statistics are written to the log file in seconds.
UsedIpAddressExpireInterval	UsedIpAddressExpireInterval *n* *time_units*	No	–1, never	Specifies how often addresses placed in the BAD state are recouped and retested for validity.

leaseExpireInterval	leaseExpireInterval *n time_units*	No	900 seconds	Specifies how often addresses in the BOUND state are checked to see if they have expired. If the address has expired, the state is moved to EXPIRED.
reservedTime	reservedTime *n time_units*	No	–1, never	Specifies how long addresses should sit in RESERVED state before being recouped into the FREE state.
reservedTimeInterval	reservedTimeInterval *n time_units*	No	900 seconds	Specifies how often addresses in the RESERVE state are checked to see if they should be recouped into the FREE state.
saveInterval	saveInterval *n time_units*	No	3600 seconds	Specifies how often the DHCP server should force a save of the open databases. For heavily loaded servers, this should be 60 or 120 seconds.
clientpruneintv	clientpruneintv *n time_units*	No	3600 seconds	Specifies how often the DHCP server has the databases remove clients are not associated with any address (in the UNKNOWN state). This reduces the memory use of the DHCP server.
numprocessors	numprocessors *n*	No	10	Specifies the number of packet processors to create. Minimum of one.

userObject	userObject *obj_name*	Yes	None	Indicates that the server should load a user–defined shared object and call routines within this object through each interaction with DHCP clients. The object to be loaded is located in the `/usr/sbin` directory by the name `obj_name.dhcpo`. See the DHCP Server User–Defined Extension API for more information.

pxeservertype	pxeservertype *server_type*	No	dhcp_only	Indicates the type of **dhcpd** server that it is. *server_type* can be one of the following: dhcp_pxe_binld DHCP performs **dhcpsd**, **pxed**, and **bindl** functions. proxy_on_dhcp_server DHCP refers the PXE client to the proxy server port on the same machine. The default is `dhcp_only`, meaning the **dhcpsd** does not support PXE clients in default mode.
supportsubnetselection	supportsubnetselection **global** supportsubnetselection **subnetlevel** supportsubnetselection **no**	No	None	Indicates whether the dhcp server will support the option 118 (subnet selection option) in the clients DISCOVER or REQUEST packet. **global**: all subnets in the configuration file will support option 118. **subnetlevel**: subnets that have been configured to support this option by keyword supportoption118 will support this option. **no**: does not support option 118.

DHCP Server File Syntax for db_file Database Notes:

1. Time Units (*time_units*) shown in the following table are optional and represent a modifier to the actual time. The default time unit is minutes. Valid values are seconds (1), minutes (60), hours (3600), days (86400), weeks (604800), months (2392000), and years (31536000). The number shown in parentheses is a multiplier applied to the specified value *n* to express the value in seconds.
2. Items that are specified in one container can be overridden inside a subcontainer. For example, you could globally define BOOTP clients, but within a certain subnet allow BOOTP clients by specifying the supportBootp keyword in both containers.
3. The client, class, and vendor containers allow for regular expression support. For class and vendor, a quoted string with the first character after the quote being an exclamation point (!) indicates that the rest of the string should be treated as a regular expression. The client container allows for regular expressions on both the hwtype and the hwaddr fields. A single string is used to represent both fields with the following format:

   ```
   decimal_number-data
   ```

   If decimal_number is zero, then data is an ASCII string. If any other number, data is hex digits.

Keyword	Form	Subcontainers?	Default Value	Meaning
subnet	subnet default	Yes	None	Specifies a subnet without an associated range. This subnet is used by the server only when responding to a client INFORM/REQUEST packet from the client and the client’s address does not have another matching subnet container.

subnet	subnet *subnet id netmask*	Yes	None	Specifies a subnet and a pool of addresses. All addresses are assumed to be in the pool unless a range is specified on the line or addresses are modified later in the container by a range or exclude statement. The optional range is a pair of IP addresses in dotted quad format separated by a dash. An optional label and priority can be specified. These are used by virtual subnets to identify and order the subnets in the virtual subnet. The label and priority are separated by a colon. These containers are only allowed at the global or database container level.
	subnet *subnet id netmask range*			
	subnet *subnet id netmask label*: *priority*			
	subnet *subnet id netmask range label*: *priority*			
subnet	subnet *subnet id range*	Yes	None	Specifies a subnet that goes within a network container. It defines a range of addresses that is the whole subnet unless the optional range part is specified. The netmask associated with the subnet is taken from the surrounding network container. **Note:** This method is deprecated in favor of the other subnet forms.

option	option *number data*...	No	None	Specifies an option to send to a client or, in the case of deny, an option to prevent from being sent to the client. The option `* deny` clause means all options not specified in the current container are not to be returned to the client. The option *number* deny only denies the specified option. *number* is an unsigned 8–bit integer. *data* is specific to the option (see above) or can be specified as a quoted string (indicating ASCII text) or 0x *hexdigits* or hex” *hexdigits* ” or hex ” *hexdigits* ”. If the option is in a vendor container, the option will be encapsulated with other options in an option 43.
	option *number* deny			
	option * deny			
exclude	exclude *an IP address*	No	None	Modifies the range on the container in which the exclude statement is in. The exclude statement is not valid in the global or database container levels. The exclude statement removes the specified address or range from the current range on the container. The exclude statement allows you to create non–contiguous ranges for subnets or other containers.
	exclude *dotted_quad – dotted_quad*			

range	range *IP_address*	No	None	Modifies the range on the container in which the range statement is in. The range statement is not valid in the global or database container levels. If the range is the first in the container that does not specify a range on the container definition line, then the range for the container becomes the range specified by the range statement. Any range statement after the first range or all range statements for a containers that specifies ranges in its definition are added to the current range. With the range statement, a single address or set of addresses can be added to the range. The range must fit inside the subnet container definition.
	range *dotted_quad – dotted_quad*			
client	client *hwtype hwaddr* NONE	Yes	None	Specifies a client container that denies the client specified by the *hwaddr* and *hwtype* from getting an address. If *hwtype* is 0, then *hwaddr* is an ASCII string. Otherwise, *hwtype* is the hardware type for the client and *hwaddr* is the hardware address of the client. If the *hwaddr* is a string, then quotes are accepted around the string. If the *hwaddr* is a hexstring, then the address may be specified by 0x *hexdigits* or hex *digits*. *range* allows the client specified by the *hwaddr* and *hwtype* to get an address in the *range*. Must be regular expressions to match multiple clients.
	client *hwtype hwaddr* ANY			
	client *hwtype hwaddr dotted_quad*			
	client *hwtype hwaddr range*			

class	class *string*	Yes	None	Specifies a class container with name *string*. String can be quoted or not. If quoted, the quotes are removed before comparison. Quotes are required for strings with spaces or tabs. This container is valid at any level. A range can be supplied to indicate a set of addresses to hand out to a client with this class. The range is either a single dotted quad IP address or two dotted quad IP addresses separated by a dash.
	class *string range*			
network	network *network id netmask*	Yes	None	Specifies a network ID using class information (for example, 9.3.149.0 with a netmask of 255.255.255.0 would be network 9.0.0.0 255.255.255.0). This version of the network container is used to hold subnets with the same network ID and netmask. When a range is provided, all the addresses in the range are in the pool. The range must be in the network ID's network. This uses class full addressing. This is only valid in the global or database container level. **Note:** The network keyword is deprecated in favor of the subnet container.
	network *network id*			
	network *network id range*			

vendor	vendor *vendor_id*	Yes	None	Specifies a vendor container. Vendor containers are used to return option 43 to the client. The vendor id may be specified in a quoted string or a binary string in the form 0x *hexdigits* or hex” *digits* ”. An optional range may be placed after the vendor id. The range is specified as two dotted quads separated by a dash. After the optional range, an optional hexstring or ASCII string can be specified as the first part of the option 43. If options are in the container, they are appended to the option 43 data. After all options are processed an End Of Option List Option is appended to the data. To return options outside of an option 43, use a regular expression client that matches all clients to specify normal options to return based on the vendor ID.
	vendor *vendor_id* hex””			
	vendor *vendor_id* hex ””			
	vendor *vendor_id* 0xdata			
	vendor *vendor_id* ””			
	vendor *vendor_id range*			
	vendor *vendor_id range* hex””			
	vendor *vendor_id range* hex ””			
	vendor *vendor_id range* 0xdata			
	vendor *vendor_id range* ””			

inoption	inoption *number option_data*	Yes	None	Specifies a container to be matched against any arbitrary incoming option specified by the client. *number* specifies the option number. *option_data* specifies the key to match for this container to be selected during address and option selection for the client. *option_data* is specified in expected form — quoted string, IP address, integer value — for well known options, or it can be optionally specified as a hexadecimal string of bytes if preceded by the characters `0x`. For options that are not well known to the server, a hexadecimal string of bytes can be specified in the same fashion. Additionally, the *option_data* can indicate a regular expression to be compared against the string representation of the client's option data. Regular expressions are specified in a quoted string beginning `″!` (double quote followed by an exclamation mark). The string form of options not well known to the server will be a hexadecimal string of bytes NOT preceded with the characters `0x`.
	inoption *number option_data range*			

virtual	virtual fill *id id*...	No	None	Specifies a virtual subnet with a policy. `fill` means use all addresses in the container before going to the next container. `rotate` means select an address from the next pool in the list on each request. `sfill` and `srotate` are the same as `fill` and `rotate`, but a search is done to see if the client matches containers, vendors, or classes in the subnet. If a match is found that can supply an address, the address is taken from that container instead of following the policy. There can be as many IDs as needed. *id* is either the subnet ID from the subnet definition or the label from the subnet definition. The label is required if there are multiple subnets with the same subnet id.
	virtual sfill *id id*...			
	virtual rotate *id id*...			
	virtual srotate *id id*...			
inorder:	inorder: *id id*...	No	None	Specifies a virtual subnet with a policy of fill, which means use all addresses in the container before going to the next container. There can be as many IDs as needed. *id* is either the subnet ID from the subnet definition or the label from the subnet definition. The label is required if there are multiple subnets with the same subnet ID.

balance:	balance: *id id*...	No	None	Specifies a virtual subnet with a policy of rotate, which means use the next address in the next container. There can be as many IDs as needed. *id* is either the subnet ID from the subnet definition or the label from the subnet definition. The label is required if there are multiple subnets with the same subnet ID.
supportBootp	supportBootp true	No	Yes	Specifies whether the current container and all below it (until overridden) should support BOOTP clients.
	supportBootp 1			
	supportBootp yes			
	supportBootp false			
	supportBootp 0			
	supportBootp no			
supportUnlistedclients	supportUnlistedclients BOTH	No	Both	Specifies whether the current container and all below it (until overridden) should support unlisted clients. The value indicates whether all clients should be allowed access without specific client statements, DHCP clients only, BOOTP clients only, or no one. **Note::** The true and false values are supported for compatibility with previous versions and are deprecated. The true value corresponds to BOTH and the false value corresponds to NONE.
	supportUnlistedclients DHCP			
	supportUnlistedclients BOOTP			
	supportUnlistedclients NONE			
	supportUnlistedclients true			
	supportUnlistedclients yes			
	supportUnlistedclients 1			
	supportUnlistedclients false			
	supportUnlistedclients no			
	supportUnlistedclients 0			
addressrecorddb	addressrecrddb *path*	No	None	If specified, it works like the **backupfile** keyword. Only valid in the global or database container level. **Note:** This method is deprecated.
backupfile	backupfile *path*	No	/etc/db_file .crbk	Specifies the file to use for database backups. Only valid in the global or database container level.

checkpointfile	checkpointfile *path*	No	/etc/db_file .chkpt	Specifies the database checkpoint files. The first checkpoint file is the *path*. The second checkpoint file is *path* with the last character replaced with a 2. So, the checkpoint file should not end in 2. Only valid in the global or database container level.
clientrecorddb	clientrecorddb *path*	No	/etc/db_file .cr	Specifies the database save file. The file contains all the client records the DHCP server has serviced. Only valid in the global or database container level.
bootstrapserver	bootstrapserver *IP address*	No	None	Specifies the server clients should use from which to TFTP files after receiving BOOTP or DHCP packets. This value fills in the **siaddr** field in the packet. This is valid at any container level.
giaddrfield	giaddrfield *IP address*	No	None	Specifies the giaddrfield for response packets. **Note:** This specification is illegal in the BOOTP and DHCP protocols, but some clients require the **giaddr** field to be the default gateway for the network. Because of this potential conflict, giaddrfield should only be used within a client container, although it can work at any level.
pingTime	pingTime *n time_unit*	No	3 seconds	Specifies the amount of time to wait for a ping response before handing out an address. The default time unit is hundredths of a second. The time unit value is defined in the note preceding this table. This is valid at any container level. The *time_unit* parameter is optional.

bootptime	bootptime *n time_unit*	No	–1, infinite	Specifies the amount of time to lease an address to a BOOTP client. The default is –1, which means infinite. The normal time unit values are available. The *time unit* parameter is optional. This is valid at any container level.
AllRoutesBroadcast	allroutesbroadcast no	No	0	Specifies whether responses should be broadcast to all routes, if a broadcast response is required. This is valid at any container level. This is ignored by the operating system’s DHCP servers, because the actual MAC address of the client, including RIFs, are stored for the return packet. This is valid at any container level.
	allroutesbroadcast false			
	allroutesbroadcast 0			
	allroutesbroadcast yes			
	allroutesbroadcast true			
	allroutesbroadcast 1			
addressassigned	addressassigned ” *string* ”	No	None	Specifies a quoted string to execute when an address is assigned to a client. The string should have two %s. The first %s is the client id in the form *type – string*. The second %s is an IP address in dotted quad format. This is valid at any container level.
addressreleased	addressreleased ” *string* ”	No	None	Specifies a quoted string to execute when an address is released by a client. The string should have one %s. The %s is the IP address being released in dotted quad format. This is valid at any container level.
appenddomain	appenddomain 0	No	No	Specifies whether to append the defined option 15 domain name to the client–suggested hostname in the event that the client does not suggest a domain name as well. This is valid at any container level.
	appenddomain no			
	appenddomain false			
	appenddomain 1			
	appenddomain yes			
	appenddomain true			

canonical	canonical 0 canonical no canonical false canonical 1 canonical yes canonical true	No	0	Specifies that the client id is in canonical format. This is valid only in the client container.
leaseTimeDefault	leaseTimeDefault *n time_unit*	No	86400 seconds	Specifies the default lease time for clients. This is valid at any container level. The *time_unit* parameter is optional.
proxyarec	proxyarec never proxyarec usedhcpddns proxyarec usedhcpddnsplus proxyarec always proxyarec usedhcpddnsprotected proxyarec usedhcpddnsplusprotected proxyarec alwaysprotected proxyarec standard proxyarec protected	No	usedhcpddnsplus	Specifies what options and methods should be used for A record updates in the DNS. `never` means never update the A record. `usedhcpddns` means use option 81 if the client specifies it. `usedhcpddnsplus` means use option 81 or option 12 and 15, if specified. `always` means do the A record update for all clients. `XXXXprotected` modifies the **nsupdate** command to make sure the client is allowed. `standard` is a synonym for `always`. `protected` is a synonym for `alwaysprotected`. This is valid at any container level.
releasednsA	releasednsA ” *string* ”	No	None	Specifies the execution string to use when an address is released. The string is used to remove the A record associated with the address released. This is valid at any container level.
releasednsP	releasednsP ” *string* ”	No	None	Specifies the execution string to use when an address is released. The string is used to remove the PTR record associated with the address released. This is valid at any container level.

removedns	removedns " *string* "	No	None	Specifies the execution string to use when an address is released. The string is used to remove the PTR and A record associated with the address released. This is valid at any container level. **Note:** This is deprecated in favor of the **releasednsA** and **releasednsP** keywords.
updatedns	updatedns " *string* "	No	None	Specifies the execution string to use when an address is bound. The string is used to update both the A and the PTR record associated with the address. This is valid at any container level. **Note:** This is deprecated in favor of the **updatednsA** and **updatednsP** keywords.
updatednsA	updatednsA " *string* "	No	None	Specifies the execution string to use when an address is bound. The string is used to update the A record associated with the address. This is valid at any container level.
updatednsP	updatednsP " *string* "	No	None	Specifies the execution string to use when an address is bound. The string is used to update the PTR record associated with the address. This is valid at any container level.

hostnamepolicy	hostnamepolicy suggested	No	default	Specifies which hostname to return to the client. Default policy is to prefer the defined hostname and domain name over suggested names. Other policies imply strict adherence (for example: `defined` will return the defined name or none if no name is defined in the configuration). Also, policies using the `always` modifier will dictate the server to return the hostname option regardless of whether the client requested it through the parameter list option. Note that suggesting a hostname also implies requesting it, and hostnames can be suggested through option 81 or through options 12 and 15. This keyword is valid at any container level.
	hostnamepolicy resolved			
	hostnamepolicy always_resolved			
	hostnamepolicy defined			
	hostnamepolicy always_defined			
	hostnamepolicy default			
bootfilepolicy	bootfilepolicy suggested	No	suggested	Specifies a preference for returning the bootfile name to a client. `suggested` prefers the client–suggested bootfile name to any server–configured name. `merge` appends the client suggested name to the server–configured home directory. `defined` prefers the defined name over any suggested bootfile name. `always` returns the defined name regardless of whether the client requests the bootfile option through the parameter list option.
	bootfilepolicy merge			
	bootfilepolicy defined			
	bootfilepolicy always			

stealfromchildren	stealfromchildren true	No	No	Specifies whether the parent container should ”steal” from children containers when the parent container runs out of addresses. This means that if you have a subnet with class defined with a range of addresses, those addresses are reserved for those clients that specify that class. If `stealfromchildren` is true, then addresses will be pulled from a child to try and satisfy the request. The default is to not steal an address.
	stealfromchildren 1			
	stealfromchildren yes			
	stealfromchildren false			
	stealfromchildren 0			
	stealfromchildren no			
homedirectory	homedirectory *path*	No	None	Specifies the home directory to use in the file section of the response packet. This can be specified at any container level. The bootfile policy defines how items specified in the file section of the incoming packet interact with the bootfile and the home directory statements.
bootfile	bootfile *path*	No	None	Specifies the bootfile to use in the file section of the response packet. This can be specified at any container level. The bootfile policy defines how items specified in the file section of the incoming packet interact with the bootfile and the home directory statements.

pxebootfile	pxebootfile *system_architecture major_version minor_version boofilename*	No	None	Specifies the bootfile to be given for a client. This is used only when **dhcpsd** supports PXE clients (**pxeservertype** is **dhcp_pxe_binld**). The configuration file parser generates an error if the number of parameters after `pxebootfile` is less than four, and it ignores any additional parameters. `pxebootfile` can only be used within a container.
supportoption118	supportoption118 *no* / *yes*	No. Can be defined only in subnet container.	None	This keyword specifies whether this container supports the option 118. Yes mean it is supported, and No means it is not supported. For this option to take effect, you have to also use the keyword **supportsubnetselection**.

DHCP and Network Installation Management (NIM) Suggestions

The concept of dynamically assigning Internet Protocol (IP) addresses is fairly new. The following suggestions are provided to help with DHCP and NIM interaction.

1. When configuring objects in the NIM environment, use host names whenever possible. This allows you to use a dynamic name server that updates the IP addresses when the host name is converted to an IP address in the NIM environment.
2. Place the NIM master and the DHCP server on the same system. The DHCP server has an option in the update DNS string that, when set to `NIM`, attempts to keep the NIM objects out of those states that need static IP addresses when those addresses change.
3. For NIM clients, set the default lease time to twice the time it takes to install a client. This allows a leased IP address to be valid during the installation. After the installation, restart the clien. DHCP will be started or will need to be configured, depending on the type of installation.
4. The **dhcpsd** server should be responsible for both the PTR and the A DNS records. When NIM reinstalls the machine, the file containing the RSA is deleted, and the client cannot update its records. The server updates the system records. To do this, change the `updatedns` line in **/etc/dhcpcd.ini** to:

```
updatedns ″/usr/sbin/dhcpaction ′%s′ ′%s′ ′%s′ ′%s′ NONE NONIM″
```

In the **/etc/dhcpsd.cnf** file, change the `updatedns` line to:

```
updatedns ″/usr/sbin/dhcpaction ′%s′ ′%s′ ′%s′ ′%s′ BOTH NIM″
```

Note:: When a NIM object is placed into the BOS installation–pending state, the **dhcpsd** server might pass arguments that are different from those originally intended. Minimize the time the client is in this pending state to avoid this situation.

These suggestions allow the NIM environment to work with dynamic clients.

For more information on Network Installation Management, see *AIX 5L Version 5.2 Network Installation Management Guide and Reference*.

Preboot Execution Environment Proxy DHCP Daemon (pxed)

The PXE Proxy DHCP server behaves much like a DHCP server by listening for ordinary DHCP client traffic and responding to certain client requests. However, unlike the DHCP server, the PXE Proxy DHCP server does not administer network addresses, and it only responds to clients that identify themselves as PXE clients. The responses given by the PXE Proxy DHCP server contain the mechanism by which the client locate sthe boot servers or the network addresses and descriptions of the supported, compatible boot servers.

Using a PXE Proxy DHCP server in addition to a DHCP server provides three key features. First, you can separate the administration of network addresses from the administration of boot images. Using two different processes on the same system, you can configure the boot information managed by the PXE Proxy DHCP server without disturbing or requiring access to the DHCP server configuration. Second, you can define multiple boot servers and let the PXE client select a particular server during boot time. Each boot server can, for example, offer a different type of operating system or system configuration. Finally, using the proxy server offers the ability to configure the PXE client to use multicast IP addressing to discover the location of the compatible boot servers.

The PXE Proxy DHCP server can be configured to run on the same system that is running the DHCP server or on a different system. Also, it can be configured to run on the same system that is also running the boot server daemon or on a different system.

The PXE Proxy DHCP Server

The PXED server is segmented into three main pieces, a database, a protocol engine, and a set of service threads, each with its own configuration information.

The PXED Database

The **db_file.dhcpo** database is used to generate the options to be sent to the client when the client send an REQUEST packet. The options returned by the database depend on the type of server chosen. This is set using the keyword **pxeservertype** in the **pxed.cnf** file.

Using the information in the configuration file, the database is primed and verified for consistency.

The PXED Protocol Engine

For AIX 4.3.1 and later, the PXED protocol engine is based on Intel’s Preboot Execution Environment (PXE) Specification Version 2.1, but is still compatible with Intel’s PXE Specification Version 1.1. The protocol engine uses the database to determine what information should be returned to the client.

PXED Threaded Operations

The last piece of the PXED server is actually a set of operations that are used to keep things running. Since the PXED server is threaded, these operations are actually set up as threads that occasionally do things to make sure everything is together.

The first thread, the *main* thread, handles the SRC requests (such as **startsrc**, **stopsrc**, **lssrc**, **traceson**, and **refresh**). This thread also coordinates all operations that affect all threads and handles signals. For example,

- A SIGHUP (–1) causes a refresh of all databases in the configuration file.
- A SIGTERM (–15) causes the server to gracefully stop.

The other thread processes packets. Depending on the server type, there can one or two threads. One thread listens on port 67 and the second one listens to port 4011. Each of these can handle a request from a client.

}

Configuring the PXED Server

By default, the PXED server is configured by reading the **/etc/pxed.cnf** file, which specifies the server's initial database of options and addresses. The server is started from the Web–based System Manager, from SMIT, or through SRC commands.

Configuring the PXED server is usually the hardest part of using PXED in your network. First, figure out what networks you need to have PXE clients on. The following example configures the **pxed** daemon to run on the same machine as the DHCP server:

```
pxeservertype       proxy_on_dhcp_server

 subnet default
 {
     vendor pxe
     {
         option    6      2       # Disable Multicast boot server discovery
         option    8      1   2     9.3.4.5    9.3.4.6   2    1
9.3.149.29
                                  # The above option gives the list of
bootservers
         option    9      0       ”PXE bootstrap server” \
                          1       ”Microsoft Windows NT Boot Server”  \
                          2       ”DOS/UNDI Boot Server”
         option    10    20    ”seconds left before the first item in the
boot menu is auto–selected”
      }
 }
```

The suboptions in the vendor container are sent to PXE clients only if the client's IP address is in the subnet's IP address range (for example, 9.3.149.0 through 9.3.149.255).

The following example configures the **pxed** daemon to run on a different machine than the DHCP server:

```
subnet default
 {
     vendor pxe
     {
         option    6      10      # The bootfile name is present in the
client’s initial pxed
                                  # offer packet.
         option    8      1   2     9.3.4.5    9.3.4.6   2    1
9.3.149.29
                                  # The above option gives the list of
bootservers
         option    9      0       ”PXE bootstrap server” \
                          1       ”Microsoft Windows NT Boot Server”  \
                          2       ”DOS/UNDI Boot Server”
         option    10    20    ”seconds left before the first item in the
boot menu is auto–selected”
         bootstrapserver          9.3.148.65
         pxebootfile    1    2    1   window.one
         pxebootfile    2    2    1   linux.one
         pxebootfile    1    2    1   hello.one
         client 6 10005a8ad14d any
         {
            pxebootfile    1    2    1   aix.one
            pxebootfile    2    2    1   window.one
         }
     }

 Vendor pxeserver
 {
     option     7     224.234.202.202
 }
```

The **pxeservertype** keyword is not set in the configuration file so the default value is taken, which is **pdhcp_only**, meaning the PXED server is running on a different machine than the DHCP server. Given this configuration, the PXED server listens on two ports (67 and 4011) for clients' BINLD REQUEST/INFORM packets. The option 7 is sent to the BINLD server when the PXED server receives a REQUEST/INFORM packet on port 67 from BINLD and option 60 is set to PXED server.

The `db_file` database clause indicates which database method to use for processing this part of the configuration file. Comments begin with a pound sign (#). From the # to the end of the line are ignored by the PXED server. Each `option` line is used by the server to tell the client what to do. PXE Vendor Container Suboptions describes the currently supported and known options. See PXED Server File Syntax for General Server Operation for ways to specify options that the server does not know about.

The Configuration File

The configuration file has an address section and an option definition section, which are based on the concept of containers that hold options, modifiers, and, potentially, other containers.

A *container* (basically, a method to group options) uses an identifier to classify clients into groups. The container types are subnet, class, vendor, and client. Currently, there is not a generic user–definable container. The identifier uniquely defines the client so that the client can be tracked if, for example, it moves between subnets. More than one container type can be used to define client access.

Options are identifiers that are returned to the client, such as default gateway and DNS address.

Containers

When the DHCP server receives a request, the packet is parsed and identifying keys determine which containers, options, and addresses are extracted

.

The previous example shows a subnet container. Its identifying key is the client's position in the network. If the client is from that network, then it falls into that container.

Each type of container uses a different option to identify a client:

- The subnet container uses the `giaddr` field or the interface address of the receiving interface to determine which subnet the client came from.
- The class container uses the value in option 77 (User Site Class Identifier).
- The vendor uses the value in option 60 (Vendor Class Identifier).
- The client container uses the option 61 (Client Identifier) for PXE clients and the `chaddr` field in the BOOTP packet for BOOTP clients.

Except for subnets, each container allows the specification of the value that it will match including regular expression matching.

There is also an implicit container, the *global* container. Options and modifiers in the global container apply to all containers unless overridden or denied. Most containers can be placed inside other containers implying a scope of visibility. Containers might or might not have address ranges associated with them. Subnets, by their nature, have ranges associated with them.

The basic rules for containers and subcontainers are as follows:

- All containers are valid at the global level.
- Subnets can never be placed inside other containers.
- Restricted containers cannot have regular containers of the same type within them. (For example, a container with an option that only allows a class of `Accounting` cannot include a container with an option that allows all classes that start with the letter "a." This is illegal.)
- Restricted client containers cannot have subcontainers.

Given the above rules, you can generate a hierarchy of containers that segment your options into groups for specific clients or sets of clients.

If a client matches multiple containers, how are options and addresses handed out? The DHCP server receives messages, it passes the request to the database (**db_file** in this case), and a container list is generated. The list is presented in order of depth and priority. Priority is defined as an implicit hierarchy in the containers. Strict containers are higher priority than regular containers. Clients, classes, vendors, and finally subnets are sorted, in that order, and within container type by depth. This generates a list ordered by most specific to least specific. For example:

```
Subnet 1
 --Class 1
 --Client 1
 Subnet 2
 --Class 1
 ----Vendor 1
 ----Client 1
 --Client 1
```

The above example shows two subnets, `Subnet 1` and `Subnet 2`. There is one class name, `Class 1`, one vendor name, `Vendor 1`, and one client name, `Client 1`. `Class 1` and `Client 1` are defined in multiple places. Because they are in different containers, their names can be the same but values inside them can be different. If `Client 1` sends a message to the DHCP server from `Subnet 1` with `Class 1` specified in its option list, the DHCP server would generate the following container path:

`Subnet 1, Class 1, Client 1`

The most specific container is listed last. To get an address, the list is examined in reverse hierarchy to find the first available address. Then, the list is examined in forward hierarchy to get the options. Options override previous values unless an option *deny* is present in the container. Also, since `Class 1` and `Client 1` are in `Subnet 1`, they are ordered according to the container priority. If the same client is in `Subnet 2` and sends the same message, the container list generated is:

`Subnet 2, Class 1, Client 1` (at the `Subnet 2` level), `Client 1` (at the `Class 1` level)

`Subnet 2` is listed first, then `Class 1`, then the `Client 1` at the `Subnet 2` level (because this client statement is only one level down in the hierarchy). The hierarchy implies that a client matching the first client statement is less specific than the client matching `Client 1` of `Class 1` within `Subnet 2`.

Priority selected by depth within the hierarchy is not superseded by the priority of the containers themselves. For example, if the same client issues the same message and specifies a vendor identifier, the container list is:

`Subnet 2, Class 1, Vendor 1, Client 1` (at `Subnet 2` level), `Client 1` (at `Class 1` level)

Container priority improves search performance because it follows a general concept that client containers are the most specific way to define one or more clients. The class container holds less specific addresses than a client container; vendor is even less specific; and subnet is the least specific.

Addresses and Address Ranges

Any container type can have associated addresses ranges; subnets must have. Each range within a container must be a subset of the parent container's range and must not overlap with other containers' ranges. For example, if a class is defined within a subnet and the class has a range, the range must be a subset of the subnet's range. Also, the range within that class container cannot overlap with any other ranges at its level.

Ranges can be expressed on the container line and modified by range and exclude statements to allow for disjoint address sets associated with a container. So, if you have the top ten addresses and the second ten addresses of a subnet available, the subnet could specify these addresses by range in the subnet clause to reduce both memory use and the chance of address collision with other clients not in the specified ranges.

Once an address has been selected, any subsequent container in the list that contains address ranges is removed from the list along with its children. The reason for this is that network–specific options in removed containers are not valid if an address is not used from within that container.

Options

After the list has been culled to determine addresses, a set of options is generated for the client. In this selection process, options overwrite previously selected options unless a *deny* is encountered, in which case, the denied option is removed from the list being sent to the client. This method allows inheritance from parent containers to reduce the amount of data that must be specified.

Logging

Logging parameters are specified in a container like the database, but the container keyword is **logging_info**. When learning to configure PXED, it is advisable to turn logging to its highest level. Also, it is best to specify the logging configuration prior to any other configuration file data to ensure that configuration errors are logged after the logging subsystem is initialized. Use the **logitem** keyword to turn on a logging level or remove the **logitem** keyword to disable a logging level. Other keywords for logging allow the specification of the log filename, file size, and the number of rotating log files.

Performance Considerations

It is important to understand that certain configuration keywords and the structure of the configuration file have an effect on the memory use and performance of the PXED server.

First, excessive memory use can be avoided by understanding the inheritance model of options from parent to child containers. In an environment that supports no unlisted clients, the administrator must explicitly list each client in the file. When options are listed for any specific client, the server uses more memory storing that configuration tree than when options are inherited from a parent container (for example, the subnet, network, or global containers). Therefore, the administrator should verify whether any options are repeated at the client level within the configuration file and, if so, determine whether these options can be specified in the parent container and shared by the set of clients as a whole.

Also, when using the **logItem** entries INFO and TRACE, numerous messages are logged during the processing of every PXE client's message. Appending a line to the log file can be an expensive operation; therefore, limiting the amount of logging improves the performance of the PXED server. When an error with the PXED server is suspected, logging can be dynamically re–enabled using the SRC traceson command.

PXE Vendor Container Suboptions

When supporting a PXE client, the DHCP server passes the following option to the BINLD server that BINLD uses to configure itself:

Opt Num	Default Data Type	Can Specify?	Description
6	Decimal number	Yes	PXE_DISCOVERY_CONTROL. Limit 0–16. This is a bit field. Bit 0 is the least significant bit. **bit 0** If set, disables broadcast discovery. **bit 1** If set, disables multicast discovery. **bit 2** If set, only uses/accepts servers in PXE_BOOT_ SERVERS. **bit 3** If set, and a bootfile name is present in the intial PXED offer packet, downloads the bootfile (does not prompt/menu/discover boot server). **bit 4–7** Must be 0. If this option is not supplied then client assumes all bits to be equal to 0.
7	One dotted quad	Yes	Multicast IP address. Boot server discovery multicast IP address. Boot servers capable of multicast discovery must listen on this multicast address. This option is required if the multicast discovery disable bit (bit 1) in the PXE_DISCOVERY_ CONTROL option is not set.

8	*Boot server type(0–65535)*	Yes	PXE_BOOT_SERVERS *IP address count (0–256)* **Type 0** Miscrosoft Windows *IP address...IP address* NT Boot Server *Boot server type IP address* **Type 1** Intel LCM Boot Server *count IP address ...* **Type 3** DOS/UNDI Boot Server *IP address* **Type 4** NEC ESMPRO Boot Server **Type 5** WSoD Boot Server **Type 6** LCCM Boot Server **Type 7** CA Unicenter TNG Boot Server. **Type 8** HP OpenView Boot Server. **Type 9 through 32767** Reserved **Type 32768 through 65534** Vendor use **Type 65535** PXE API Test Server. If *IP address count* is zero for a server type then the client may accept offers from any boot server of that type. Boot Servers do not respond to discovery requests of types they do not support.
9	*Boot server type (0–65535)*	Yes	PXE_BOOT_MENU *"description"* Boot server boot "order" is implicit in the type. *"description"...* *menu order.*
10	*Timeout in seconds (0–255)*	Yes	PXE_MENU_PROMPT *"prompt"* The timeout is the number of seconds to wait before auto–selecting the first boot menu item. On the client system, the prompt is displayed followed by the number of seconds remaining before the first item in the boot menu is auto–selected. If the F8 key is pressed on the client system, then a menu is displayed. If this option is provided to the client, then the menu is displayed without prompt and timeout. If the timeout is 0, then the first item in the menu is auto–selected. If the timeout is 255, the menu and prompt is displayed without auto–selecting or timeout.

PXED Server File Syntax for General Server Operation

Note:: Time Units (*time_units*) shown in the following table are optional and represent a modifier to the actual time. The default time unit is minutes. Valid values are seconds (1), minutes (60), hours (3600), days (86400), weeks (604800), months (2392000), and years (31536000). The number shown in parentheses is a multiplier applied to the specified value *n* to express the value in seconds.

Keyword	Form	Subcontainers?	Default Value	Meaning
database	database *db type*	Yes	None	The primary container that holds the definitions for the address pools, options, and client access statements. *db type* is the name of a module that is loaded to process this part of the file. The only value currently available is **db_file**.
logging_info	logging_info	Yes	None	The primary logging container that defines the logging parameters.
logitem	logitem NONE logitem SYSERR logitem OBJERR logitem PROTOCOL logitem PROTERR logitem WARN logitem WARNING logitem CONFIG logitem EVENT logitem PARSEERR logitem ACTION logitem ACNTING logitem STAT logitem TRACE logitem RTRACE logitem START	No	All default to not enabled.	Enables the logging level. Multiple lines are allowed.
numLogFiles	numLogFiles *n*	No	0	Specifies the number of log files to create. The log rotates when the first one fills. *n* is the number of files to create.

logFileSize	logFileSize *n*	No	0	Specifies the size of each log file in 1024–byte units.
logFileName	logFileName *path*	No	None	Specifies the path to the first log file. The original log file is named *filename* or *filename.extension*. The *filename* must be eight or fewer characters. When a file is rotated, it is renamed beginning with the base *filename*, then either appending a number or replacing the extension with a number. For example, if the original file name is `file`, the rotated file name becomes `file01`. If the original file name is `file.log`, it becomes `file.01`.
pxeservertype	pxeservertype *servertype*	No	dhcp_only	Indicates the type of **dhcpsd** server it is. *servertype* can be **proxy_on_dhcp_server**, which means that PXED is running on the same machine as the DHCP server and it is listening for PXE client requests on port 4011 only, or the default value of **pdhcp_only**, which means the PXED is running on a separate machine and it has to listen for client packets on port 67 and 4011.

PXED Server File Syntax for db_file Database Notes:

1. Time Units (*time_units*) shown in the following table are optional and represent a modifier to the actual time. The default time unit is minutes. Valid values are seconds (1), minutes (60), hours (3600), days (86400), weeks (604800), months (2392000), and years (31536000). The number shown in parentheses is a multiplier applied to the specified value *n* to express the value in seconds.
2. Items that are specified in one container can be overridden inside a subcontainer. For example, you could globally define BOOTP clients, but within a certain subnet allow BOOTP clients by specifying the supportBootp keywork in both containers.

3. The client, class, and vendor containers allow for regular expression support. For class and vendor, a quoted string with the first character after the quote being an exclamation point (!) indicates that the rest of the string should be treated as a regular expression. The client container allows for regular expressions on both the hwtype and the hwaddr fields. A single string is used to represent both fields with the following format:

```
decimal_number–data
```

If decimal_number is zero, then data is an ASCII string. If any other number, data is hex digits.

Keyword	**Form**	**Subcontainers?**	**Default Value**	**Meaning**
subnet	subnet default	Yes	None	Specifies a subnet that does not have any range. The subnet is used by the server only when it is responding to INFORM packet from the client.
subnet	subnet *subnet id netmask*	Yes	None	Specifies a subnet and a pool of addresses. All addresses are assumed to be in the pool unless a range is specified on the line or addresses are modified later in the container by a range or exclude statement. The optional range is a pair of IP addresses in dotted quad format separated by a dash. An optional label and priority can be specified. These are used by virtual subnets to identify and order the subnets in the virtual subnet. The label and priority are separated by a colon. These containers are only allowed at the global or database container level.
	subnet *subnet id netmask range*			
	subnet *subnet id netmask label*: *priority*			
	subnet *subnet id netmask range label*: *priority*			

subnet	subnet *subnet id range*	Yes	None	Specifies a subnet that goes within a network container. It defines a range of addresses that is the whole subnet unless the optional range part is specified. The netmask associated with the subnet is taken from the surrounding network container. **Note::** This method is deprecated in favor of the other subnet forms.
option	option *number data*...	No	None	Specifies an option to send to a client or, in the case of deny, an option to prevent from being sent to the client. The optional * `deny` clause means all options not specified in the current container are not to be returned to the client. option *number* deny only denies the specified option. *number* is an unsigned 8–bit integer. *data* is specific to the option (see above) or can be specified as a quoted string (indicating ASCII text) or 0x *hexdigits* or hex” *hexdigits* ” or hex ” *hexdigits* ”. If the option is in a vendor container, the option will be encapsulated with other options in an option 43.
	option *number* deny			
	option * deny			

exclude	exclude *an IP address*	No	None	Modifies the range on the container in which the exclude statement is in. The exclude statement is not valid in the global or database container levels. The exclude statement removes the specified address or range from the current range on the container. The exclude statement allows you to create non–contiguous ranges for subnets or other containers.
	exclude *dotted_quad – dotted_quad*			
range	range *IP_address*	No	None	Modifies the range on the container in which the range statement is in. The range statement is not valid in the global or database container levels. If the range is the first in the container that does not specify a range on the container definition line, then the range for the container becomes the range specified by the range statement. Any range statement after the first range or all range statements for a containers that specifies ranges in its definition are added to the current range. With the range statement, a single address or set of addresses can be added to the range. The range must fit inside the subnet container definition.
	range *dotted_quad – dotted_quad*			

client	client *hwtype hwaddr* NONE	Yes	None	Specifies a client container that denies the client specified by the *hwaddr* and *hwtype* from getting an address. If *hwtype* is 0, then *hwaddr* is an ASCII string. Otherwise, *hwtype* is the hardware type for the client and *hwaddr* is the hardware address of the client. If the *hwaddr* is a string, then quotes are accepted around the string. If the *hwaddr* is a hexstring, then the address may be specified by 0x *hexdigits* or hex *digits*. *range* allows the client specified by the *hwaddr* and *hwtype* to get an address in the *range*. Must be regular expressions to match multiple clients.
	client *hwtype hwaddr* ANY			
	client *hwtype hwaddr dotted_quad*			
	client *hwtype hwaddr range*			
class	class *string*	Yes	None	Specifies a class container with name *string*. String can be quoted or not. If quoted, the quotes are removed before comparison. Quotes are required for strings with spaces or tabs. This container is valid at any level. A range can be supplied to indicate a set of addresses to hand out to a client with this class. The range is either a single dotted quad IP address or two dotted quad IP addresses separated by a dash.
	class *string range*			

network	network *network id netmask*	Yes	None	Specifies a network ID using class information (for example, 9.3.149.0 with a netmask of 255.255.255.0 would be network 9.0.0.0 255.255.255.0). This version of the network container is used to hold subnets with the same network ID and netmask. When a range is provided, all the addresses in the range are in the pool. The range must be in the network ID's network. This uses class full addressing. This is only valid in the global or database container level. **Note::** The network keyword is deprecated in favor of the subnet container.
	network *network id*			
	network *network id range*			

vendor	vendor *vendor_id*	Yes	None	Specifies a vendor container. Vendor containers are used to return option 43 to the client. The vendor id may be specified in a quoted string or a binary string in the form 0x *hexdigits* or hex” *digits* ”. An optional range may be placed after the vendor id. The range is specified as two dotted quads separated by a dash. After the optional range, an optional hexstring or ASCII string can be specified as the first part of the option 43. If options are in the container, they are appended to the option 43 data. After all options are processed an End Of Option List Option is appended to the data. To return options outside of an option 43, use a regular expression client that matches all clients to specify normal options to return based on the vendor ID.
	vendor *vendor_id* hex””			
	vendor *vendor_id* hex ””			
	vendor *vendor_id* 0xdata			
	vendor *vendor_id* ””			
	vendor *vendor_id* *range*			
	vendor *vendor_id* *range* hex””			
	vendor *vendor_id* *range* hex ””			
	vendor *vendor_id* *range* 0xdata			
	vendor *vendor_id* *range* ””			

inoption	inoption *number option_data*	Yes	None	Specifies a container to be matched against any arbitrary incoming option specified by the client. *number* specifies the option number. *option_data* specifies the key to match for this container to be selected during address and option selectoin for the client. *option_data* is specified in expected form — quoted string, IP address, integer value — for well known options, or it can be optionally speicifed as a hexadecimal string of bytes if preceded by the characters `0x`. For options that are not well known to the server, a hexadecimal string of bytes can be specified in the same fashion. Additionally, the *option_data* can indicate a regular expression to be compared against the string representation of the client's option data. Regular expressions are specified in a quoted string beginning `"!` (double quote followed by an exclamation mark). The string form of options not well known to the server will be a hexadecimal string of bytes NOT preceded with the characters `0x`.
	inoption *number option_data range*			

virtual	virtual fill *id id*...	No	None	Specifies a virtual subnet with a policy. `fill` means use all addresses in the container before going to the next container. `rotate` means select an address from the next pool in the list on each request. `sfill` and `srotate` are the same as `fill` and `rotate`, but a search is done to see if the client matches containers, vendors, or classes in the subnet. If a match is found that can supply an address, the address is taken from that container instead of following the policy. There can be as many IDs as needed. *id* is either the subnet ID from the subnet definition or the label from the subnet definition. The label is required if there are multiple subnets with the same subnet id.
	virtual sfill *id id*...			
	virtual rotate *id id*...			
	virtual srotate *id id*...			
inorder:	inorder: *id id*...	No	None	Specifies a virtual subnet with a policy of fill, which means use all addresses in the container before going to the next container. There can be as many IDs as needed. *id* is either the subnet ID from the subnet definition or the label from the subnet definition. The label is required if there are multiple subnets with the same subnet ID.

balance:	balance: *id id*...	No	None	Specifies a virtual subnet with a policy of rotate, which means use the next address in the next container. There can be as many IDs as needed. *id* is either the subnet ID from the subnet definition or the label from the subnet definition. The label is required if there are multiple subnets with the same subnet ID.
bootstrapserver	bootstrapserver *IP address*	No	None	Specifies the server clients should use from which to TFTP files after receiving BOOTP or DHCP packets. This value fills in the **siaddr** field in the packet. This is valid at any container level.
giaddrfield	giaddrfield *IP address*	No	None	Specifies the giaddrfield for response packets. **Note::** This specification is illegal in the BOOTP and DHCP protocols, but some clients require the **giaddr** field to be the default gateway for the network. Because of this potential conflict, giaddrfield should only be used within a client container, although it can work at any level.

bootfile	bootfile *path*	No	None	Specifies the bootfile to use in the file section of the response packet. This can be specified at any container level. The bootfile policy defines how items specified in the file section of the incoming packet interact with the bootfile and the home directory statements.
pxebootfile	pxebootfile *System Arch MajorVer MinorVer Bootfilename*	No	None	Specifies the bootfile to be given to a client. The config file parser generates an error if the number of parameters after the keyword is less than 4 and ignore if more than 4. This keyword can be used only in a container.

For details about other options, see DHCP Server File Known Options.

Boot Image Negotiation Layer Daemon (BINLD)

The Boot Image Image Negotiation Layer daemon (BINLD) server is the third stage of contact for preboot execution environment (PXE) clients. After communicating with the DHCP server to obtain an IP address, and after communication with the PXE Proxy DHCP server to obtain the location of the boot server, the boot server is contacted to get the filename and location from which to download the boot image. The PXE client can return to communicate with the boot server multiple times in the course of booting if the client requires multiple files in its boot process.

The final stage in the PXE network boot is to download the boot image given by the boot server. The location of the TFTP server and the filename that is to be downloaded is given by the boot server to the PXE client.

The BINLD Server

Beginning with AIX 4.3.3 update, the BINLD server is segmented into three main pieces: a database, a protocol engine, and a set of service threads, each with its own configuration information.

The BINLD Database

The **db_file.dhcpo** database is used to generate the options that respond to a client's REQUEST packet. The options returned by the database depend on the type of server chosen. Options are set using the keyword **pxeservertype** in the **binld.cnf** file.

Using the information in the configuration file, the database is primed and verified for consistency.

The BINLD Protocol Engine

The PXED protocol engine is based on the Intel's Preboot Execution Environment (PXE) Specification Version 2.1, but is still compatible with Intel's PXE Specification Version 1.1. The protocol engine uses the database to determine what information should be returned to the client.

BINLD Threaded Operations

The last piece of the BINLD server is actually a set of operations that are used to keep things running. Since the BINLD server is threaded, these operations are actually set up as threads that occasionally do things to make sure everything is together.

The first thread, the *main* thread, handles the SRC requests (such as **startsrc**, **stopsrc**, **lssrc**, **traceson**, and **refresh**). This thread also coordinates all operations that affect all threads and handles signals. For example,

- A SIGHUP (–1) causes a refresh of all databases in the configuration file.
- A SIGTERM (–15) causes the server to gracefully stop.

The other thread processes packets. Depending on the server type, there can one or two threads. One thread listens on port 67 and the second to port 4011. Each can handle a request from a client.

Configuring BINLD

By default, the BINLD server is configured by reading the **/etc/binld.cnf** file, which specifies the server's initial database of options and addresses. The server is started from the Web–based System Manager, from SMIT, or through SRC commands.

Configuring the BINLD server is usually the hardest part of using BINLD in your network. First, figure out what networks you need to have PXE clients on. The following example configures a BINLD server to run on the same machine as the DHCP server:

```
pxeservertype       binld_on_dhcp_server

 subnet default
 {
     vendor pxe
     {
                 bootstrapserver  9.3.149.6      #TFTP server IP address
                 pxebootfile   1   2   1   window.one   1   0
                 pxebootfile   2   2   1   linux.one    2   3
                 pxebootfile   1   2   1   hello.one    3   4
                 client 6 10005a8ad14d any
                 {
                    pxebootfile   1   2   1   aix.one     5   6
                    pxebootfile   2   2   1   window.one  6   7
         }
      }
 }
```

Given the above configuration, the BINLD server listens for client's unicast packets on port 4011 and Multicast packets on port 4011 if BINLD gets the Multicast Address from the dhcpsd/pxed. The BINLD server responds to client REQUEST/INFORM packets with the bootfile name and TFTP server's IP address. If BINLD does not find the bootfile with a matching Layer specified by the client, then it tries to find a bootfile for the next layer. The BINLD does not respond when there is no boot file that matches the client requirements (*Type*, *SystemArch*, *MajorVers*, *MinorVers*, and *Layer*).

The following example configures BINLD to run on a separate machine (that is, DHCP / PXED is not running on the same machine).

```
subnet 9.3.149.0 255.255.255.0
 {

      vendor pxe
      {

      bootstrapserver        9.3.149.6      # TFTP server ip address.
      pxebootfile    1    2    1   window.one    1   0
      pxebootfile    2    2    1   linux.one     2   3
      pxebootfile    1    2    1   hello.one     3   4
      client 6 10005a8ad14d any
        {
        pxebootfile    1    2    1   aix.one      5   6
        pxebootfile    2    2    1   window.one   6   7
        }
      }
 }
```

In the above example, the *pxeservertype* is not set, so the default servertype is **binld_only**. The BINLD server listens for client's unicast packets on port 4011, broadcast & unicast packets on port 67, and Multicast packets on port 4011 if BINLD gets the Multicast Address from the dhcpsd/pxed. The bootfile name and TFTP server IP address is sent to a PXE client only if the client's IP address is in the subnet's IP address range (9.3.149.0 through 9.3.149.255).

The following example configures BINLD to run on the same machine as the PXED server:

```
pxeservertype       binld_on_proxy_server
 subnet default
 {
      vendor
      {
      bootstrapserver        9.3.149.6          # TFTP server ip address.
      pxebootfile    1    2    1   window.one    1    0
      pxebootfile    2    2    1   linux.one     2    3
      pxebootfile    1    2    1   hello.one     3    4
      client 6 10005a8ad14d any
        {
        pxebootfile    1    2    1   aix.one     5   6
        pxebootfile    2    2    1   window.one  6   7
        }
      }
 }
```

In this configuraton, the BINLD server only listens on port 4011 for Multicast packets only if BINLD gets Multicast address from the dhcpsd/pxed. If it does not receive any multicast address, then BINLD exits and an error message is logged to the log file.

The database `db_file` clause indicates which database method to use for processing this part of the configuration file. Comments begin with a pound sign (#). From the # to the end of the line are ignored by the PXED server. Each `option` line is used by the server to tell the client what to do. PXE Vendor Container Suboptions describes the currently supported and known suboptions. See BINLD Server File Syntax for General Server Operation for ways to specify options that the server does not know about.

The Configuration File

The configuration file has an address section and an option definition section, which are based on the concept of containers that hold options, modifiers, and, potentially, other containers.

A *container* (basically, a method to group options) uses an identifier to classify clients into groups. The container types are subnet, class, vendor, and client. Currently, there is not a generic user–definable container. The identifier uniquely defines the client so that the client can be tracked if, for example, it moves between subnets. More than one container type can be used to define client access.

Options are identifiers that are returned to the client, such as default gateway and DNS address.

Containers

When the DHCP server receives a request, the packet is parsed and identifying keys determine which containers, options, and addresses are extracted.

The previous example shows a subnet container. Its identifying key is the client's position in the network. If the client is from that network, then it falls into that container.

Each type of container uses a different option to identify a client:

- The subnet container uses the giaddr field or the interface address of the receiving interface to determine which subnet the client came from.
- The class container uses the value in option 77 (User Site Class Identifier).
- The vendor uses the value in option 60 (Vendor Class Identifier).
- The client container uses the option 61 (Client Identifier) for PXED clients and the chaddr field in the BOOTP packet for BOOTP clients.

Except for subnets, each container allows the specification of the value that it matches, including regular expression matching.

There is also an implicit container, the *global* container. Options and modifiers placed in the global container apply to all containers unless overridden or denied. Most containers can be placed inside other containers implying a scope of visibility. Containers may or may not have address ranges associated with them. Subnets, by their nature, have ranges associated with them.

The basic rules for containers and subcontainers are as follows:

- All containers are valid at the global level.
- Subnets can never be placed inside other containers.
- Restricted containers cannot have regular containers of the same type within them. (For example, a container with an option that only allows a class of `Accounting` cannot include a container with an option that allows all classes that start with the letter ”a”. This is illegal.)
- Restricted client containers cannot have subcontainers.

Given the above rules, you can generate a hierarchy of containers that segment your options into groups for specific clients or sets of clients.

If a client matches multiple containers, how are options and addresses handed out? The DHCP server receives messages, it passes the request to the database (**db_file** in this case), and a container list is generated. The list is presented in order of depth and priority. Priority is defined as an implicit hierarchy in the containers. Strict containers are higher priority than regular containers. Clients, classes, vendors, and finally subnets are sorted, in that order, and within container type by depth. This generates a list ordered by most specific to least specific. For example:

```
Subnet 1
 --Class 1
 --Client 1
 Subnet 2
 --Class 1
 ----Vendor 1
 ----Client 1
 --Client 1
```

The example shows two subnets, `Subnet 1` and `Subnet 2`. There is one class name, `Class 1`, one vendor name, `Vendor 1`, and one client name, `Client 1`. `Class 1` and `Client 1` are defined in multiple places. Because they are in different containers, their names can be the same but values inside them may be different. If `Client 1` sends a message to the DHCP server from `Subnet 1` with `Class 1` specified in its option list, the DHCP server would generate the following container path:

`Subnet 1, Class 1, Client 1`

The most specific container is listed last. To get an address, the list is examined in reverse hierarchy to find the first available address. Then, the list is examined in forward hierarchy to get the options. Options override previous values unless an option *deny* is present in the container. Also, since `Class 1` and `Client 1` are in `Subnet 1`, they are ordered according to the container priority. If the same client is in `Subnet 2` and sends the same message, the container list generated is:

`Subnet 2, Class 1, Client 1` (at the `Subnet 2` level), `Client 1` (at the `Class 1` level)

`Subnet 2` is listed first, then `Class 1`, then the `Client 1` at the `Subnet 2` level (because this client statement is only one level down in the hierarchy). The hierarchy implies that a client matching the first client statement is less specific than the client matching `Client 1` of `Class 1` within `Subnet 2`.

Priority selected by depth within the hierarchy is not superseded by the priority of the containers themselves. For example, if the same client issues the same message and specifies a vendor identifier, the container list is:

`Subnet 2, Class 1, Vendor 1, Client 1` (at `Subnet 2` level), `Client 1` (at `Class 1` level)

Container priority improves search performance because it follows a general concept that client containers are the most specific way to define one or more clients. The class container holds less specific addresses than a client container; vendor is even less specific; and subnet is the least specific.

Addresses and Address Ranges

Any container type may have associated addresses ranges; subnets must have. Each range within a container must be a subset of the parent container's range and must not overlap with other containers' ranges. For example, if a class is defined within a subnet and the class has a range, the range must be a subset of the subnet's range. Also, the range within that class container cannot overlap with any other ranges at its level.

Ranges can be expressed on the container line and modified by range and exclude statements to allow for disjoint address sets associated with a container. So, if you have the top ten addresses and the second ten addresses of a subnet available, the subnet could specify these addresses by range in the subnet clause to reduce both memory use and the chance of address collision with other clients not in the specified ranges.

Once an address has been selected, any subsequent container in the list that contains address ranges is removed from the list along with its children. The reason for this is that network–specific options in removed containers are not valid if an address is not used from within that container.

Options

After the list has been culled to determine addresses, a set of options is generated for the client. In this selection process, options overwrite previously selected options unless a *deny* is encountered, in which case, the denied option is removed from the list being sent to the client. This method allows inheritance from parent containers to reduce the amount of data that must be specified.

Logging

Logging parameters are specified in a container like the database, but the container keyword is **logging_info**. When learning to configure PXED, it is advisable to turn logging to its highest level. Also, it is best to specify the logging configuration prior to any other configuration file data to ensure that configuration errors are logged after the logging subsystem is initialized. Use the **logitem** keyword to turn on a logging level or remove the **logitem** keyword to disable a logging level. Other keywords for logging allow the specification of the log filename, file size, and the number of rotating log files.

Performance Considerations

It is important to understand that certain configuration keywords and the structure of the configuration file have an effect on the memory use and performance of the PXED server.

First, excessive memory use can be avoided by understanding the inheritance model of options from parent to child containers. In an environment that supports no unlisted clients, the administrator must explicitly list each client in the file. When options are listed for any specific client, the server uses more memory storing that configuration tree than when options are inherited from a parent container (for example, the subnet, network, or global containers). Therefore, the administrator should verify whether any options are repeated at the client level within the configuration file and, if so, determine whether these options can be specified in the parent container and shared by the set of clients as a whole.

Also, when using the **logItem** entries INFO and TRACE, numerous messages are logged during the processing of every PXE client's message. Appending a line to the log file can be an expensive operation; therefore, limiting the amount of logging improves the performance of the PXED server.When an error with the PXED server is suspected, logging can be dynamically re–enabled using the SRC traceson command.

BINLD Server File Syntax for General Server Operation

Note:: Time Units (*time_units*) shown in the following table are optional and represent a modifier to the actual time. The default time unit is minutes. Valid values are seconds (1), minutes (60), hours (3600), days (86400), weeks (604800), months (2392000), and years (31536000). The number shown in parentheses is a multiplier applied to the specified value *n* to express the value in seconds.

Keyword	**Form**	**Subcontainers?**	**Default Value**	**Meaning**
database	database *db type*	Yes	None	The primary container that holds the definitions for the address pools, options, and client access statements. *db type* is the name of a module that is loaded to process this part of the file. The only value currently available is **db_file**.
logging_info	logging_info	Yes	None	The primary logging container that defines the logging parameters.
logitem	logitem NONE	No	All default to not enabled.	Enables the logging level. Multiple lines are allowed.
	logitem SYSERR			
	logitem OBJERR			
	logitem PROTOCOL			
	logitem PROTERR			
	logitem WARN			
	logitem WARNING			
	logitem CONFIG			
	logitem EVENT			
	logitem PARSEERR			
	logitem ACTION			
	logitem ACNTING			
	logitem STAT			
	logitem TRACE			
	logitem RTRACE			
	logitem START			

numLogFiles	numLogFiles *n*	No	0	Specifies the number of log files to create. The log rotates when the first one fills. *n* is the number of files to create.
logFileSize	logFileSize *n*	No	0	Specifies the size of each log file in 1024–byte units.
logFileName	logFileName *path*	No	None	Specifies the path to the first log file. The original log file is named *filename* or *filename.extension*. The *filename* must be eight or fewer characters. When a file is rotated, it is renamed beginning with the base *filename*, then either appending a number or replacing the extension with a number. For example, if the original file name is `file`, the rotated file name becomes `file01`. If the original file name is `file.log`, it becomes `file.01`.

pxeservertype	pxeservertype *servertype*	No	dhcp_only	Indicate the type of dhcpsd server it is. *servertype* can be one of the following **binld_on_dhcp_server** This means that **BINLD** is running on the same machine as DHCP server and it is listening for PXE Client request on port 4011 and Multicast address if received from the DHCP / PXED. **binld_on_proxy_server** This means that **BINLD** is running on the same machine as PXED server and it is listening for PXE Client's request on Multicast address if received from the DHCP / PXED. The default value is **binld_only** ie the BINLD is running on a separate machine and it has to listen for client's packets on port 67 , 4011 and Multicast address if received from the DHCP / PXED.
dhcp_or_proxy_address	dhcp_or_proxy_address *IP address*	No	None	This gives the IP address of dhcp or pxed server to which the BINLD server can send an Unicast packet of type REQUEST/INFORM to receive the Multicast Address. This keyword is defined only when the dhcp or pxed are on a different subnet than BINLD.

BINLD Server File Syntax for db_file Database Notes:

1. Time Units (*time_units*) shown in the following table are optional and represent a modifier to the actual time. The default time unit is minutes. Valid values are seconds (1), minutes (60), hours (3600), days (86400), weeks (604800), months (2392000), and years (31536000). The number shown in parentheses is a multiplier applied to the specified value *n* to express the value in seconds.
2. Items that are specified in one container can be overridden inside a subcontainer. For example, you could globally define BOOTP clients, but within a certain subnet allow BOOTP clients by specifying the supportBootp keyword in both containers.
3. The client, class, and vendor containers allow for regular expression support. For class and vendor, a quoted string with the first character after the quote being an exclamation point (!) indicates that the rest of the string should be treated as a regular expression. The client container allows for regular expressions on both the hwtype and the hwaddr fields. A single string is used to represent both fields with the following format:

```
decimal_number-data
```

If decimal_number is zero, then data is an ASCII string. If any other number, data is hex digits.

Keyword	Form	Subcontainers?	Default Value	Meaning
subnet	subnet default	Yes	None	Specifies a subnet that does not have any range. The subnet is used by a server only when it is responding to INFORM packet from the client and the client's address does not have another matching subnet container.
subnet	subnet *subnet id netmask*	Yes	None	Specifies a subnet and a pool of addresses. All addresses are assumed to be in the pool unless a range is specified on the line or addresses are modified later in the container by a range or exclude statement. The optional range is a pair of IP addresses in dotted quad format separated by a dash. An optional label and priority can be specified. These are used by virtual subnets to identify and order the subnets in the virtual subnet. The label and priority are separated by a colon. These containers are only allowed at the global or database container level.
	subnet *subnet id netmask range*			
	subnet *subnet id netmask label*: *priority*			
	subnet *subnet id netmask range label*: *priority*			

subnet	subnet *subnet id range*	Yes	None	Specifies a subnet that goes within a network container. It defines a range of addresses that is the whole subnet unless the optional range part is specified. The netmask associated with the subnet is taken from the surrounding network container. **Note::** This method is deprecated in favor of the other subnet forms
option	option *number data*...	No	None	Specifies an option to send to a client or, in the case of deny, an option to prevent from being sent to the client. The option `* deny` clause means all options not specified in the current container are not to be returned to the client. option *number* deny only denies the specified option. *number* is an unsigned 8–bit integer. *data* is specific to the option (see above) or can be specified as a quoted string (indicating ASCII text) or 0x *hexdigits* or hex” *hexdigits* ” or hex ” *hexdigits* ”. If the option is in a vendor container, the option will be encapsulated with other options in an option 43.
	option *number* deny			
	option * deny			
exclude	exclude *an IP address*	No	None	Modifies the range on the container in which the exclude statement is in. The exclude statement is not valid in the global or database container levels. The exclude statement removes the specified address or range from the current range on the container. The exclude statement allows you to create non–contiguous ranges for subnets or other containers.
	exclude *dotted_quad* – *dotted_quad*			

range	range *IP_address*	No	None	Modifies the range on the container in which the range statement is in. The range statement is not valid in the global or database container levels. If the range is the first in the container that does not specify a range on the container definition line, then the range for the container becomes the range specified by the range statement. Any range statement after the first range or all range statements for a containers that specifies ranges in its definition are added to the current range. With the range statement, a single address or set of addresses can be added to the range. The range must fit inside the subnet container definition.
	range *dotted_quad* – *dotted_quad*			
client	client *hwtype hwaddr* NONE	Yes	None	Specifies a client container that denies the client specified by the *hwaddr* and *hwtype* from getting an address. If *hwtype* is 0, then *hwaddr* is an ASCII string. Otherwise, *hwtype* is the hardware type for the client and *hwaddr* is the hardware address of the client. If the *hwaddr* is a string, then quotes are accepted around the string. If the *hwaddr* is a hexstring, then the address may be specified by 0x *hexdigits* or hex *digits*. *range* allows the client specified by the *hwaddr* and *hwtype* to get an address in the *range*. Must be regular expressions to match multiple clients.
	client *hwtype hwaddr* ANY			
	client *hwtype hwaddr dotted_quad*			
	client *hwtype hwaddr range*			

class	class *string*	Yes	None	Specifies a class container with name *string*. String can be quoted or not. If quoted, the quotes are removed before comparison. Quotes are required for strings with spaces or tabs. This container is valid at any level. A range can be supplied to indicate a set of addresses to hand out to a client with this class. The range is either a single dotted quad IP address or two dotted quad IP addresses separated by a dash.
	class *string range*			
network	network *network id netmask*	Yes	None	Specifies a network ID using class information (for example, 9.3.149.0 with a netmask of 255.255.255.0 would be network 9.0.0.0 255.255.255.0). This version of the network container is used to hold subnets with the same network ID and netmask. When a range is provided, all the addresses in the range are in the pool. The range must be in the network ID's network. This uses class full addressing. This is only valid in the global or database container level. **Note::** The network keyword is deprecated in favor of the subnet container.
	network *network id*			
	network *network id range*			

vendor	vendor *vendor_id*	Yes	None	Specifies a vendor container. Vendor containers are used to return option 43 to the client. The vendor id may be specified in a quoted string or a binary string in the form 0x *hexdigits* or hex” *digits* ”. An optional range may be placed after the vendor id. The range is specified as two dotted quads separated by a dash. After the optional range, an optional hexstring or ASCII string can be specified as the first part of the option 43. If options are in the container, they are appended to the option 43 data. After all options are processed an End Of Option List Option is appended to the data. To return options outside of an option 43, use a regular expression client that matches all clients to specify normal options to return based on the vendor ID. **pxe** after the keyword **vendor** will create a vendor container for PXEClient. **pxeserver** after the keyword **vendor** will create a vendor container for PXEServer.
	vendor *vendor_id* hex””			
	vendor *vendor_id* hex ””			
	vendor *vendor_id* 0xdata			
	vendor *vendor_id* ””			
	vendor *vendor_id range*			
	vendor *vendor_id range* hex””			
	vendor *vendor_id range* hex ””			
	vendor *vendor_id range* 0xdata			
	vendor *vendor_id range* ””			
	vendor pxe			
	vendor pxeserver			

inoption	inoption *number option_data*	Yes	None	Specifies a container to be matched against any arbitrary incoming option specified by the client. *number* specifies the option number. *option_data* specifies the key to match for this container to be selected during address and option selection for the client. *option_data* is specified in expected form — quoted string, IP address, integer value — for well known options, or it can be optionally specified as a hexadecimal string of bytes if preceded by the characters `0x`. For options that are not well known to the server, a hexadecimal string of bytes can be specified in the same fashion. Additionally, the *option_data* can indicate a regular expression to be compared against the string representation of the client's option data. Regular expressions are specified in a quoted string beginning `″!` (double quote followed by an exclamation mark). The string form of options not well known to the server will be a hexadecimal string of bytes NOT preceded with the characters `0x`.
	inoption *number option_data range*			

virtual	virtual fill *id id*...	No	None	Specifies a virtual subnet with a policy. `fill` means use all addresses in the container before going to the next container. `rotate` means select an address from the next pool in the list on each request. `sfill` and `srotate` are the same as `fill` and `rotate`, but a search is done to see if the client matches containers, vendors, or classes in the subnet. If a match is found that can supply an address, the address is taken from that container instead of following the policy. There can be as many IDs as needed. *id* is either the subnet ID from the subnet definition or the label from the subnet definition. The label is required if there are multiple subnets with the same subnet id.
	virtual sfill *id id*...			
	virtual rotate *id id*...			
	virtual srotate *id id*...			
inorder:	inorder: *id id*...	No	None	Specifies a virtual subnet with a policy of fill, which means use all addresses in the container before going to the next container. There can be as many IDs as needed. *id* is either the subnet ID from the subnet definition or the label from the subnet definition. The label is required if there are multiple subnets with the same subnet ID.

balance:	balance: *id id*...	No	None	Specifies a virtual subnet with a policy of rotate, which means use the next address in the next container. There can be as many IDs as needed. *id* is either the subnet ID from the subnet definition or the label from the subnet definition. The label is required if there are multiple subnets with the same subnet ID.
bootstrapserver	bootstrapserver *IP address*	No	None	Specifies the server clients should use from which to TFTP files after receiving BOOTP or DHCP packets. This value fills in the **siaddr** field in the packet. This is valid at any container level.
giaddrfield	giaddrfield *IP address*	No	None	Specifies the giaddrfield for response packets. **Note::** This specification is illegal in the BOOTP and DHCP protocols, but some clients require the **giaddr** field to be the default gateway for the network. Because of this potential conflict, giaddrfield should only be used within a client container, although it can work at any level.

bootfile	bootfile *path*	No	None	Specifies the bootfile to use in the file section of the response packet. This can be specified at any container level. The bootfile policy defines how items specified in the file section of the incoming packet interact with the bootfile and the home directory statements.
pxebootfile	pxebootfile *SystemArch MajorVer MinorVer Bootfilename Type Layer*	No	None	Specifies the bootfile to be given to a PXEClient. The config file parser generates an error if the number of parameters after the keyword is less than 4 , ignore if more than 7 and if 4 are there then it assume the value for Type = 0 and Layer = 0. This keyword can be used only in a container.

For details about other options, see DHCP Server File Known Options on page 4-107 and PXE Vendor Container Suboptions on page 4-143.

TCP/IP Daemons

Daemons (also known as *servers*) are processes that run continuously in the background and perform functions required by other processes. Transmission Control Protocol/Internet Protocol (TCP/IP) provides daemons for implementing certain functions in the operating system. These daemons are background processes that run without interrupting other processes (unless that is part of the daemon function).

Daemons are invoked by commands at the system management level, by other daemons, or by shell scripts. You can also control daemons with the **inetd** daemon, the **rc.tcpip** shell script, and the System Resource Controller (SRC).

Subsystems and Subservers

A *subsystem* is a daemon, or server, that is controlled by the SRC. A *subserver* is a daemon that is controlled by a subsystem. (Daemon commands and daemon names are usually denoted by a **d** at the end of the name.) The categories of subsystem and subserver are mutually exclusive. That is, daemons are not listed as both a subsystem and as a subserver. The only TCP/IP subsystem that controls other daemons is the **inetd** daemon. All TCP/IP subservers are also **inetd** subservers.

The following are TCP/IP daemons controlled by the SRC:

Subsystems

gated Provides gateway routing functions and supports the Routing Information Protocol (RIP), the Routing Information Protocol Next Generation (RIPng), Exterior Gateway Protocol (EGP), the Border Gateway Protocol (BGP) and BGP4+, the Defense Communications Network Local–Network Protocol (HELLO), Open Shortest Path First (OSPF), Intermediate System to Intermediate System (IS–IS), and Internet Control Message Protocol (ICMP and ICMPv6)/Router Discovery routing protocols. In addition, the **gated** daemon supports the Simple Network Management Protocol (SNMP). The **gated** daemon is one of two routing daemons available for routing to network addresses and is the preferred routing daemon. The **gated** daemon is preferred over the **routed** daemon because the **gated** daemon supports more gateway protocols.

inetd Invokes and schedules other daemons when requests for the daemon services are received. This daemon can also start other daemons. The **inetd** daemon is also known as the super daemon.

iptrace Provides interface–level packet–tracing function for Internet protocols.

named Provides the naming function for the **Domain Name Server** protocol (DOMAIN).

routed Manages the network routing tables and supports the **Routing Information Protocol** (RIP). The **gated** daemon is preferred over the **routed** daemon because the **gated** daemon supports more gateway protocols.

rwhod Sends broadcasts to all other hosts every three minutes and stores information about logged–in users and network status. Use the **rwhod** daemon with extreme care, because it can use significant amounts of machine resources.

timed Provides the time server function.

Note: Both the **routed** and **gated** daemons are listed as TCP/IP subsystems. Do not run the **startsrc –g tcpip** command, which initiates both of these routing daemons, along with all the other TCP/IP subsystems. Running both daemons simultaneously on one machine can produce unpredictable results.

TCP/IP daemons controlled by the **inetd** subsystem are the following:

inetd Subservers

comsat Notifies users of incoming mail.

fingerd Provides a status report on all logged–in users and network status at the specified remote host. This daemon uses the **Finger** protocol.

ftpd Provides the file transfer function for a client process using the **File Transfer Protocol** (FTP).

rexecd Provides the foreign host server function for the **rexec** command.

rlogind Provides the remote login facility function for the **rlogin** command.

rshd Provides the remote command execution server function for the **rcp** and **rsh** commands.

talkd Provides the conversation function for the **talk** command.

syslogd Reads and logs system messages. This daemon is in the **Remote Access Service** (RAS) group of subsystems.

telnetd Provides the server function for the **TELNET** protocol.

tftpd Provides the server function for the **Trivial File Transfer Protocol** (TFTP).

uucpd Handles communications between the Basic Network Utilities (BNU) and TCP/IP.

System Resource Control (SRC)

Among other functions, SRC allows you to start daemons, stop them, and trace their activity. In addition, SRC provides the ability to group daemons into subsystems and subservers.

System Resource Control is a tool designed to aid the person who manages your system in controlling daemons. SRC allows control beyond the flags and parameters available with each daemon command.

See the System Resource Controller Overview in *AIX 5L Version 5.2 System Management Concepts: Operating System and Devices* for more information concerning the System Resource Controller.

SRC Commands

SRC commands can affect one daemon, a group of daemons, or a daemon and those daemons it controls (subsystem with subservers). In addition, some TCP/IP daemons do not respond to all SRC commands. The following is a list of SRC commands that can be used to control TCP/IP daemons and their exceptions.

startsrc Starts all TCP/IP subsystems and **inetd** subservers. The **startsrc** command works for all TCP/IP subsystems and **inetd** subservers.

stopsrc Stops all TCP/IP subsystems and **inetd** subservers. This command is also called the **stop normal** command. The **stop normal** command allows subsystems to process all outstanding work and terminate gracefully. For **inetd** subservers, all pending connections are allowed to start and all existing connections are allowed to complete. The **stop normal** command works for all TCP/IP subsystems and **inetd** subservers.

stopsrc –f
Stops all TCP/IP subsystems and **inetd** subservers. This command is also called the **stop force**. The **stop force** command immediately terminates all subsystems. For **inetd** subservers, all pending connections and existing connections are terminated immediately.

refresh
Refreshes the following subsystems and subservers: the **inetd**, **syslogd**, **named**, **dhcpsd**, and **gated** subsystems.

lssrc
Provides short status for subsystems, which is the state of the specified subsystem (active or inoperative). Also provides short status for **inetd** subservers. The short status for **inetd** subservers includes: subserver name, state, subserver description, command name, and the arguments with which it was invoked.

lssrc –l
Provides the short status plus additional information (long status) for the following subsystems:

gated
State of debug or trace, routing protocols activated, routing tables, signals accepted and their function.

inetd
State of debug, list of active subservers and their short status; signals accepted and their function.

named
State of debug, **named.conf** file information.

dhcpsd
State of debug, all controlled IP addresses and their current state.

routed
State of debug and trace, state of supplying routing information, routing tables.

syslogd
syslogd configuration information.

The **lssrc –l** command also provides long status for **inetd** subservers. The long status includes short status information and active connection information. Some subservers will provide additional information. The additional information by subserver includes:

ftpd
State of debug and logging

telnetd
Type of terminal emulating

rlogind
State of debug

fingerd
State of debug and logging

The **rwhod** and **timed** subservers do not provide long status.

traceson
Turns on socket–level debugging. Use the **trpt** command to format the output. The **timed** and **iptraced** subsystems do not support the **traceson** command.

tracesoff
Turns off socket–level debugging. Use the **trpt** command to format the output. The **timed** and **iptraced** subsystems do not support the **tracesoff** command.

For examples of how to use these commands, see the articles on the individual commands. For more information on the System Resource Controller, see System Resource Controller

Overview in *AIX 5L Version 5.2 System Management Concepts: Operating System and Devices*.

Configuring the inetd Daemon

To configure the **inetd** daemon:

1. Specify which subservers it will be invokde by adding an **inetd** daemon.
2. Specify the restart characteristics by changing the restart characteristics of the **inetd** daemon.

Configuring the inetd daemon tasks

Task	**SMIT fast path**	**Command or file**	Web-based System Manager Management Environment
Starting the **inetd** Daemon	**smit mkinetd**	**startsrc –s inetd**	Software —> **Network** —> **TCPIP (IPv4 and IPv6)** —> **Subsystems**. Right–click on an inactive subsystem, and select **Activate**.
Changing Restart Characteristics of the **inetd** Daemon	**smit chinetd** or **smit lsinetd**		Software —> **Network** —> **TCPIP (IPv4 and IPv6)** —> **Subsystems** —> **Selected** —> **Properties**.
Stopping the **inetd** Daemon	**smit rminetd**	**stopsrc –s inetd**	Software —> **Network** —> **TCPIP (IPv4 and IPv6)** —> **Subsystems**. Right–click on an active subsystem, and select —> **Deactivate**.
Listing All **inetd** Subservers	**smit inetdconf**		Software —> **Network** —> **TCPIP (IPv4 and IPv6)** —> **Subsystems**.
Adding an **inetd** Subserver [1]	**smit mkinetdconf**	edit **/etc/inetd.conf** then run **refresh –s inetd** or **kill –1** *inetdPID* [2]	Software —> **Network** —> **TCPIP (IPv4 and IPv6)** —> **Subsystems** —> **Subsystems** (drop–down menu) —> **New inetd Subserver**.

Change/Show Characteristics of an **inetd** Subserver	**smit inetdconf**	edit **/etc/inetd.conf** then run **refresh –s inetd** or **kill –1** *inetdPID* 2	Software —> **Network** —> **TCPIP (IPv4 and IPv6)** —> **Subsystems** —> **Selected** —> **Properties**.
Removing an **inetd** Subserver	**smit rminetd**	edit **/etc/inetd.conf** then run **refresh –s inetd** or **kill –1** *inetdPID* 2	Software —> **Network** —> **TCPIP (IPv4 and IPv6)** —> **Subsystems** —> **Selected** —> **Deactivate**.

Notes:

1. Adding an **inetd** subserver configures the **inetd** daemon so that it invokes the subserver when it is needed.
2. Both the **refresh** and the **kill** commands inform the **inetd** daemon of changes to its configuration file.

Client Network Services

Client Network Services (accessible using the Web-based System Manager **wsm**, or the SMIT fast path, **smit clientnet**) refers to the TCP/IP protocols available for use by this operating system. Each protocol (or service) is known by the port number that it uses on the network, hence the term **well–known port**. As a convenience to programmers, the port numbers can be referred to by names as well as numbers. For example, the TCP/IP mail protocol uses port 25 and is known by the name **smtp**. If a protocol is listed (uncommented) in the **/etc/services** file, then a host can use that protocol.

By default, all the TCP/IP protocols are defined in the **/etc/services** file. You do not have to configure this file. If you write your own client/server programs, you might want to add your service to the **/etc/services** file, and reserve a specific port number and name for your service. If you do decide to add your service to **/etc/services**, note that port numbers 0 through 1024 are reserved for system use.

Client network services tasks

Task	**SMIT fast path**	**Command or file**	Web-based System Manager Management Environment
Listing All Services	**smit lsservices**	view **/etc/services**	Software —> **Network** —> **TCPIP (IPv4 and IPv6)** —> **Services**.
Adding a Service	**smit mkservices**	edit **/etc/services**	Software —> **Network** —> **TCPIP (IPv4 and IPv6)** —> **Services** —> **New Service**.

Change/Show Characteristics of a Service	**smit chservices**	edit **/etc/services**	Software —> **Network** —> **TCPIP (IPv4 and IPv6)** —> **Services**. Select a service, then click **Selected** —> **Properties**.
Removing a Service	**smit rmservices**	edit **/etc/services**	Software —> **Network** —> **TCPIP (IPv4 and IPv6)** —> **Services**. Select a service, then click **Selected** —> **Delete**.

Server Network Services

Server Network Services include controlling remote access, starting or stopping TCP/IP, and managing the **pty** device driver, as shown in the following table.

The **pty** device driver is installed automatically with the system. By default, it is configured to support 16 BSD–style symbolic links, and it is available for use by the system at boot time.

Server network services tasks

Task	**SMIT fast path**	**Command or file**	Web-based System Manager Management Environment
Controlling Remote Access	See ”Remote Command Execution Access” and ”Restricted File Transfer Program Users”.		Software —> **Network** —> **TCPIP (IPv4 and IPv6)** —> **Access Control**. Right–click on **Remote Access** and select **Properties**.
Start, Restart, or Stop TCP/IP Subsystems	**smit otherserv**	See System Resource Control (SRC) on page 4-175.	Software —> **Network** —> **TCPIP (IPv4 and IPv6)** —> **Subsystems**. Right–click on a subsystem, and select **Properties**.
Change/Show Characteristics of the pty Device Driver	**smit chgpty**	**chdev –l pty0 –P –a num=** *X* where *X* ranges from 0 to 64	
Make the pty Device Driver Unavailable for Use	**smit pty** then select **Remove the PTY; Keep Definition**		
Make the pty Device Driver Available for Use	**smit pty** then select **Configure the Defined PTY**		
Generate an Error Report	**smit errpt**		
Trace the pty	**smit trace**		

TCP/IP Routing

The topics discussed in this section are:

- Static and Dynamic Routing on page 4-180
- Gateways on page 4-181
- Planning for Gateways on page 4-183
- Configuring a Gateway on page 4-184
- Restricting Route Use on page 4-186
- Dead Gateway Detection on page 4-186
- Route Cloning on page 4-187
- Manually Removing Dynamic Routes on page 4-187
- Configuring the routed Daemon on page 4-187
- Getting an Autonomous System Number on page 4-190

A *route* defines a path for sending packets through the Internet network to an address on another network. A route does not define the complete path, only the path segment from one host to a gateway that can forward packets to a destination (or from one gateway to another). There are three types of routes:

host route Defines a gateway that can forward packets to a specific host on another network.

network route Defines a gateway that can forward packets to any of the hosts on a specific network.

default route Defines a gateway to use when a host or network route to a destination is not otherwise defined.

Routes are defined in the kernel *routing table*. The route definitions include information on networks reachable from the local host and on gateways that can be used to reach remote networks. When a gateway receives a datagram, it checks the routing tables to find out where next to send the datagram along the path to its destination.

Beginning with AIX 5.1, you can add multiple routes for the same destination in the kernel routing table. A routing lookup evaluates all routes that match the request then chooses the route with the lowest distance metric. If multiple matching routes have equal distance, a lookup chooses the most specific route. If both criteria are equal for multiple routes, routing lookups alternate choices of matching routes.

Static and Dynamic Routing

In TCP/IP, routing can be one of two types: *static* or *dynamic*. With static routing, you maintain the routing table manually using the **route** command. Static routing is practical for a single network communicating with one or two other networks. However, as your network begins to communicate with more networks, the number of gateways increases, and so does the amount of time and effort required to maintain the routing table manually.

With dynamic routing, daemons update the routing table automatically. Routing daemons continuously receive information broadcast by other routing daemons, and so continuously update the routing table.

TCP/IP provides two daemons for use in dynamic routing, the **routed** and **gated** daemons. The **gated** daemon supports Routing Information Protocol (RIP), Routing Information Protocol Next Generation (RIPng), Exterior Gateway Protocol (EGP), Border Gateway Protocol (BGP) and BGP4+, Defense Communications Network Local–Network Protocol (HELLO), Open Shortest Path First (OSPF), Intermediate System to Intermediate System

(IS–IS), and Internet Control Message Protocol (ICMP and ICMPv6)/Router Discovery routing protocols simultaneously. In addition, the **gated** daemon supports the Simple Network Management Protocol (SNMP). The **routed** daemon only supports Routing Information Protocol.

Routing daemons can operate in one of two modes, *passive* or *active*, depending upon the options you use when starting the daemons. In active mode, routing daemons both broadcast routing information periodically about their local network to gateways and hosts, and receive routing information from hosts and gateways. In passive mode, routing daemons receive routing information from hosts and gateways, but do not attempt to keep remote gateways updated (they do not advertise their own routing information).

These two types of routing can be used not only for gateways, but for other hosts on a network as well. Static routing works the same for gateways as for other hosts. Dynamic routing daemons, however, must be run in the passive (quiet) mode when run on a host that is not a gateway.

Gateways

Gateways are a type of router. *Routers* connect two or more networks and provide the routing function. Some routers, for example, route at the network interface level or at the physical level.

Gateways, however, route at the network level. Gateways receive IP datagrams from other gateways or hosts for delivery to hosts on the local network, and route IP datagrams from one network to another. For example, a gateway connecting two Token–Ring networks has two Token–Ring adapter cards, each with its own Token–Ring network interface. To pass on information, the gateway receives datagrams through one network interface and sends them out through the other network interface. Gateways periodically verify their network connections through interface status messages.

Gateways route packets according to the destination network, not according to the destination host. That is, a gateway machine is not required to keep track of every possible host destination for a packet. Instead, a gateway routes packets according to the network of the destination host. The destination network then takes care of sending the packet to the destination host. Thus, a typical gateway machine requires only limited disk storage capacity (if any) and limited main memory capacity.

The distance a message must travel from originating host to destination host depends upon the number of *gateway hops* it must make. A gateway is zero hops from a network to which it is directly attached, one hop from a network that is reachable through one gateway, and so on. Message distance is usually expressed in the number of gateway hops required, or *hop counts* (also called the *metric*).

Interior and Exterior Gateways

Interior gateways are gateways that belong to the same autonomous system. They communicate with each other using the Routing Information Protocol (RIP), Routing Information Protocol Next Generation (RIPng), Intermediate System to Intermediate System protocol, Open Shortest Path First protocol (OSPF), or the HELLO Protocol (HELLO). Exterior gateways belong to different autonomous systems. They use the Exterior Gateway Protocol (EGP), the Border Gateway Protocol (BGP), or BGP4+.

For example, consider two autonomous systems. The first is all the networks administered by the Widget Company. The second is all the networks administered by the Gadget Company. The Widget Company has one machine, called apple, which is Widget’s gateway to the Internet. The Gadget Company has one machine, called orange, which is Gadget’s gateway to the Internet. Both companies have several different networks internal to the companies. The gateways connecting the internal networks are interior gateways. But apple and orange are exterior gateways.

Each exterior gateway does not communicate with every other exterior gateway. Instead, the exterior gateway acquires a set of neighbors (other exterior gateways) with which it communicates. These neighbors are not defined by geographic proximity, but rather by their established communications with each other. The neighboring gateways, in turn, have other exterior gateway neighbors. In this way, the exterior gateway routing tables are updated and routing information is propagated among the exterior gateways.

The routing information is sent in a pair, (N,D), where N is a network and D is a distance reflecting the cost of reaching the specified network. Each gateway advertises the networks it can reach and the costs of reaching them. The receiving gateway calculates the shortest paths to other networks and passes this information along to its neighbors. Thus, each exterior gateway is continually receiving routing information, updating its routing table and then passing that information to its exterior neighbors.

Gateway Protocols

All gateways, whether interior or exterior, use protocols to communicate with each other. Here are brief descriptions of the more commonly used TCP/IP gateway protocols:

HELLO Protocol (HELLO)

HELLO is one protocol that the interior gateways use to communicate among themselves. HELLO calculates the shortest path to other networks by determining the path that has the least delay time.

Routing Information Protocol (RIP)

Routing Information Protocol is a protocol that the interior gateways use to communicate among themselves. Like the HELLO Protocol, RIP calculates the shortest path to other networks. Unlike HELLO, RIP estimates distance not by delay time, but by hop counts. Because the **gated** daemon stores all metrics internally as time delays, it converts RIP hop counts into time delays.

Routing Information Protocol Next Generation

RIPng is the RIP protocol that is enhanced to support IPv6.

Open Shortest Path First (OSPF)

OPSF is a protocol that the interior gateways use to communicate among themselves. It is a link–state protocol that is bettter suited than RIP for complex networks with many routers. It provides equal cost multipath routing.

Exterior Gateway Protocol (EGP)

The exterior gateways can use the Exterior Gateway Protocol to communicate among themselves. The EGP does not calculate the shortest path to other networks. Instead, it merely indicates whether a particular network is reachable or not.

Border Gateway Protocol (BGP)

The exterior gateways can use this protocol to communicate among themselves. It exchanges reachability information between automomous systems, but provides more capabilities than EGP. BGP uses path attributes to provide more information about each route as an aid in selecting the best route.

Border Gateway Protocol 4+

BGP4+ is the BGP protocol version 4, which supports IPv6 and has other enhancements over past versions of the protocol.

Intermediate System to Intermediate System (IS–IS)

Interior gateways use IS–IS protocol to communicate among themselves. It is a link–state protocol that can route IP and ISO/CLNP packets and, like OSPF, uses a ”shorter path first” algorithm to determine routes.

Planning for Gateways

Before you configure the gateways for your network, you must first do the following:

1. Consider the number of gateways to use (see Consider the Number of Gateways to Use).
2. Decide on the type of routing to use (see Decide on the Type of Routing to Use).

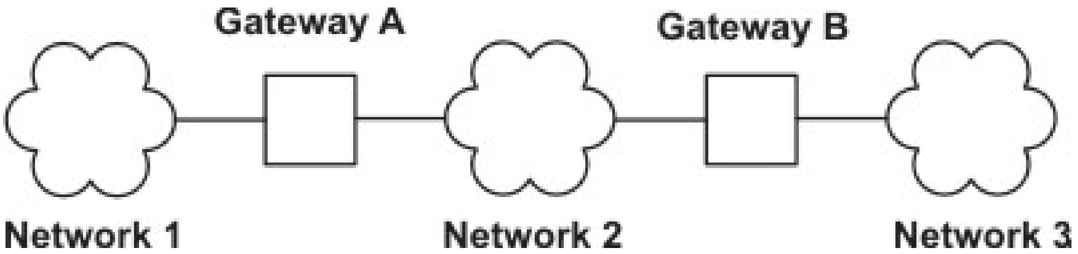
Consider the Number of Gateways to Use

The number of gateways you need to configure will depend upon:

- The number of networks you want to connect.
- How you want to connect the networks.
- The level of activity on the connected networks.

For example, suppose users on Network 1, Network 2, and Network 3 all need to communicate with each other.

Figure 25. Simple Gateway Configuration This illustration contains three network clouds numbered one, two, and three. Networks one and two are connected with gateway A. Networks two and three are connected with gateway B.



To connect Network 1 directly to Network 2, you would use a single gateway (Gateway A). To connect Network 2 directly to Network 3, you would use another gateway (Gateway B). Now, assuming the proper routes are defined, all the users on all three networks can communicate.

However, if Network 2 is very busy, communication between Network 1 and Network 3 might suffer unacceptable delays. Furthermore, if most of the inter–network communication occurs between Network 1 and Network 3, you might want to connect Network 1 directly to Network 3. To do this, you could use an additional pair of gateways, Gateway C (on Network 1) and Gateway D (on Network 3), with a direct connection between these two additional gateways. This may be an inefficient solution, however, because one gateway can connect more than two networks.

A more efficient solution would be to connect Gateway A to Gateway B directly, as well as to Network 2. This would require a second network adapter in both Gateway A and Gateway B. In general, the number of networks you connect through a single gateway is limited by the number of network adapter cards the gateway machine can support.

Decide on the Type of Routing to Use

If your network is small, and its configuration rarely changes, you probably want to use static routing. But if you have a large network whose configuration changes frequently, you probably want to use dynamic routing. You might decide to use a combination of static and dynamic routing. That is, you might want to give static definitions to a few specific routes, while allowing other routes to be updated by the daemons. The static routes you create are not advertised to other gateways and are not updated by the routing daemons.

If Using Dynamic Routing

Choose the routing daemon according to the type of gateway you need and the protocols your gateway must support. If the gateway is an interior gateway, and only needs to support

RIP, choose the **routed** daemon. If the gateway must support any other protocol, or is an exterior gateway, choose the **gated** daemon.

Note:: Unpredictable results can occur if the **gated** and **routed** daemons run on the same host at the same time.

Configuring a Gateway

To configure a machine to act as a gateway, use the following instructions. For clarity, this procedure assumes that the gateway machine connects two networks, and that the gateway machine has already been minimally configured on one of the networks.

1. Install and configure the second network adapter, if you have not done so already. (See Installing a Network Adapter on page 4-36 and Configuring and Managing Adapters on page 4-36.)
2. Choose an IP address for the second network interface, and then configure the network interface by following the instructions in Managing Network Interfaces on page 4-54.
3. Add a route to the second network.
4. To use a machine as an internetwork router over TCP/IP networks, enter:

```
no –o ipforwarding=1
```

5. The gateway machine can now access both of the networks to which it is directly attached.
   a. If you want to use static routing to communicate with hosts or networks beyond these two networks, add any other routes you want.
   b. If you want to use dynamic routing, follow the instructions in either Configuring the routed Daemon on page 4-187 or Configuring the gated Daemon on page 4-188. If your internetwork is to join the Internet, you should also follow the instructions in Getting an Autonomous System Number on page 4-190.

Configuring gateway tasks

Task	**SMIT fast path**	**Command file**	Web-based System Manager Management Environment
Displaying the Routing Table	**smit lsroute**	**netstat –rn** [1]	Software —> **Network** —> **TCPIP (IPv4 and IPv6)** —> **TCPIP Protocol Configuration** —> **TCP/IP** —> **Configure TCP/IP** —> **Advanced Methods** —> **Static Routes** —> **Statistics**.

Adding a Static Route	**smit mkroute**	**route add** *destination gateway* [2]	Software —> **Network** —> **TCPIP (IPv4 and IPv6)** —> **TCPIP Protocol Configuration** —> **TCP/IP** —> **Configure TCP/IP** —> **Advanced Methods** —> **Static Routes**. Complete the following in **Add/Change a static route**: **Destination Type**, **Gateway address**, **Network interface name** (drop–down menu), **Subnet mask**, **Metric (Cost)**, and the **Enable active dead gateway detection** check box. Click **Add/Change Route**.
Removing a Static Route	**smit rmroute**	**route delete** *destination gateway* [2]	Software —> **Network** —> **TCPIP (IPv4 and IPv6)** —> **TCPIP Protocol Configuration** —> **TCP/IP** —> **Configure TCP/IP** —> **Advanced Methods** —> **Static Routes**. Select a route, and click **Delete Route**.
Flushing the Routing Table	**smit fshrttbl**	**route flush**	Software —> **Network** —> **TCPIP (IPv4 and IPv6)** —> **TCPIP Protocol Configuration** —> **TCP/IP** —> **Configure TCP/IP** —> **Advanced Methods** —> **Static Routes** —> **Delete All**.

Notes:

1. The table is divided into columns for destination address, gateway address, flags, reference count (hop count), and network interface. (For a detailed discussion of each of these columns, see the **netstat** command in the *AIX 5L Version 5.2 Commands Reference.*) If frames are not reaching their destination and the routing tables indicate the correct route, one or more of the following conditions might exist:

   - Network is failing.

   - Remote host or gateway is failing.

- Remote host or gateway is down or not ready to receive frames.
- Remote host does not have a route back to the source network.

2. The *destination* value is the dotted decimal address or symbolic name of the destination host or network, and the *gateway* value is the dotted decimal address or symbolic name of the gateway. (A default route specifies 0 as the destination.)

Restricting Route Use

Routes can be restricted so they can be used only by some users. The restrictions are based on the primary group IDs of users. Using the **route** command, you can specify a list of up to 32 group IDs that are allowed or not allowed to use a route. If the list is of allowed groups, any user that belongs to any group on the list can use the route. If the list is of disallowed groups, only users that do not belong to any of the groups on the list can use the route. The root user can use any route.

Groups can also be associated with an interface using the **ifconfig** command. In this case, a forwardable packet can use any route allowed for the groups associated with its incoming interface.

If there are two or more routes to the same destination, any ICMP redirects that are received for that destination will be ignored and path MTU discovery will not be done on those routes.

Dead Gateway Detection

In AIX 5.1 and later, a host can be configured to detect whether a gateway it is using is down, and can adjust its routing table accordingly. If the network option **–passive_dgd** is 1, passive dead gateway detection is enabled for the entire system. If no response is received for consecutive **dgd_packets_lost** ARP requests to a gateway, that gateway is assumed to be down and the distance metrics (also known as *hopcount* or *cost*) for all routes using that gateway are raised to the maximum possible value. After **dgd_retry_time** minutes have passed, the route's costs are restored to their user–configured values. The host also takes action based on failing TCP connections. If consecutive **dgd_packets_lost** TCP packets are lost, the ARP entry for the gateway in use is deleted and the TCP connection tries the next–best route. The next time the gateway is used, the above actions take place if the gateway is actually down. The **passive_dgd**, **dgd_packets_lost**, and **dgd_retry_time** parameters can all be configured using the **no** command.

Hosts can also be configured to use active dead gateway detection on a per–route basis with the **–active_dgd** flag of the **route** command. Active dead gateway detection pings all gateways used by routes for which it is enabled every **dgd_ping_time** seconds. If no response is received from a gateway, it is pinged more rapidly up to **dgd_packets_lost** times. If still no response is received, the costs of all routes using that gateway are raised. The gateway continues to be pinged, and if a response is eventually received, the costs on the routes are restored to their user–configured values. The **dgd_ping_time** parameter can be configured using the **no** command.

Dead gateway detection is generally most useful for hosts that use static rather than dynamic routing. Passive dead gateway detection has low overhead and is recommended for use on any network that has redundant gateways. However, passive dead gateway detection is done on a best–effort basis only. Some protocols, such as UDP, do not provide any feedback to the host if a data transmission is failing, and in this case no action can be taken by passive dead gateway detection.

Active dead gateway detection is most useful when a host must discover immediately when a gateway goes down. Since it queries each gateway for which it is enabled every few seconds, there is some network overhead associated with its use. Active dead gateway detection is recommended only for hosts that provide critical services and on networks with a limited number of hosts.

Note: Dead Gateway Detection and the routing protocols used by the **gated** and **routed** daemons perform a similar function by discovering changes in the network configuration and adjusting the routing table accordingly. However, they use different mechanisms to do this, and if they are run at the same time, they may conflict with one another. For this reason, Dead Gateway Detection should not be used on systems running the **gated** or **routed** daemons.

Route Cloning

Route Cloning allows a host route to be created for every host that a system communicates with. When network traffic is about to be sent, a search is done of the routing table to find a route to that host. If a specific host route is found, it will be used. If a specific host route is not found, a network route or the default route may be found. If the route that is found has the cloning flag, 'c', set, a host route for the destination will be created using the gateway from the route that is being cloned. Subsequent routing table searches for that destination will find the cloned host route. Cloned routes have the 'W' flag set. These routes will time out and be deleted from the routing table if they are unused for **route_expire** minutes. You can modify **route_expire** by using the no command.

The route cloning feature is used mainly by the path MTU discovery protocol within AIX to allow it to keep track of path MTU information for every destination it communicates with. If the network options **tcp_pmtu_discover** or **udp_pmtu_discover** (settable with the no command) are 1, the cloning flag is turned on for all network routes on the system. In AIX 4.3.3 and above, path MTU discovery is on by default.

Manually Removing Dynamic Routes

If you are using the **routed** daemon, a manually deleted route is *not* replaced by incoming RIP information (because ioctl's are used). If you are using the **gated** daemon, and the –n flag is not used, the manually deleted route *is* replaced by the route as discovered in incoming RIP information.

Configuring the routed Daemon

To configure the **routed** daemon:

1. Remove the comment symbol (#) and modify the routed clause in the **/etc/rc.tcpip** shell script. This automatically starts the **routed** daemon with each system startup.

   – Specify whether you want the gateway to run in active (**–s** flag) or passive (**–q** flag) mode.

   – Specify whether you want packet tracing on or off (**–t** flag). Packet tracing can also be turned on after the **routed** daemon is already started by using the **kill** command to send a **SIGUSR1** signal to the daemon. This signal can also be used to increment the level of tracing through four levels. In addition, packet tracing can be turned off while the **routed** daemon is running by using the **kill** command to send a **SIGUSR2** signal to the daemon. For more information, see the **routed** daemon and the **kill** command.

   – Specify whether you want debugging turned on or off (**–d** flag). If you use this flag, specify which log file you want debugging information stored in, or choose for it to be directed to the console display.

   – Specify whether you are running the **routed** daemon on a gateway (**–g** flag).

   Note:: A host that is not a gateway can run the **routed** daemon, but it must be run in passive mode.

2. Identify any known networks by listing them in the **/etc/networks** file. See Networks File Format for TCP/IP in the *AIX 5L Version 5.2 Files Reference* for more information. A sample **networks** file is located in the **/usr/samples/tcpip** directory.

3. Set up routes in the **/etc/gateways** file to any known gateways that are not directly connected to your network. Refer to Gateways File Format for TCP/IP in *AIX 5L Version*

5.2 Files Reference for detailed examples of entries in the **/etc/gateways** file. A sample **gateways** file is located in the **/usr/samples/tcpip** directory. **Attention:** Do not run the **routed** daemon and the **gated** daemon on the same machine. Unpredictable results can occur.

Configuring the gated Daemon

To configure the **gated** daemon:

1. Decide which gateway protocols are most appropriate for your system. The choices for routing protocols are EGP, BGP, RIP, RIPng, HELLO, OSPF, ICMP/Router Discovery, and IS–IS. You can also use SNMP, a protocol allowing you to change or show management information for a network element from a remote host.

   Note:: Use EGP, BGP, or BGP4+ to advertise addresses of networks in an autonomous system to gateways in other autonomous systems. If you are on the Internet, EGP, BGP, or BGP4+ must be used to advertise network reachability to the core gateway system. Use the interior routing protocols to advertise reachability information within an autonomous system.

2. Identify any known networks by listing them in the **/etc/networks** file. See Networks File Format for TCP/IP in *AIX 5L Version 5.2 Files Reference* for more information. A sample **networks** file is located in the **/usr/samples/tcpip** directory.

3. Edit the **/etc/gated.conf** file to reflect the desired **gated** daemon configuration.

   Note:: The **gated** version on AIX 4.3.2 and higher is 3.5.9. The syntax of the **/etc/gated.conf** file has changed. The examples given below are for the 3.5.9 version of **gated**. To configure the **/etc/gated.conf** file for versions prior to AIX 4.3.2, use the syntax provided in the **/etc/gated.conf** file itself.

   a. Specify the level of trace output you want. If tracing is needed before the **gated.conf** file is parsed, use the **–t** flag to turn tracing on when the daemon starts. See gated Daemon in *AIX 5L Version 5.2 Commands Reference* for more information.

   b. Specify the routing protocols you want to use. Each protocol has its own protocol statement. Remove the comment symbols (#) and modify the statements corresponding to the protocols you want to use.

      . If using EGP:

      – Set up the EGP `autonomoussystem` clause. Obtain an autonomous system number from the Internet authority if you are on the Internet, or if not, assign an autonomous system number considering the autonomous system numbers of other systems on your network.

      – Set the EGP statement to `yes`.

      – Set up a `group` clause for each autonomous system.

      – Set up a `neighbor` clause for each neighbor in that autonomous system. For example:

```
autonomoussystem 283 ;

 egp yes {
         group maxup 1 {
              neighbor nogendefault 192.9.201.1 ;
              neighbor nogendefault 192.9.201.2 ;
        } ;
        group {
              neighbor 192.10.201.1 ;
              neighbor 192.10.201.2 ;
        } ;
 } ;
```

- If using RIP or HELLO:
  - Set the RIP or HELLO statement to yes.
  - Specify nobroadcast in the RIP or HELLO statement if you want the gateway only to accept routing information, not broadcast information. Or specify broadcast in the RIP or HELLO statement if you want the gateway to broadcast routing information as well as accept routing information.
  - If you want the gateway to send directly to source gateways, use the sourcegateways statement. Specify a gateway name or Internet address in dotted decimal in the sourcegateways clause. For example:

```
# Send directly to specific gateways

 rip/hello yes {
           sourcegateways
                 101.25.32.1
                 101.25.32.2 ;
 } ;
```

    The following example shows the RIP/HELLO stanza in the **gated.conf** file of a machine that does not send RIP packets, and does not receive RIP packets on its tr0 interface.

```
rip/hello nobroadcast {
          interface tr0 noripin ;
 } ;
```

- If using BGP:
  - Set up the BGP autonomoussystem clause. Obtain an autonomous system number from the Internet authority if you are on the Internet, or if not, assign an autonomous system number considering the autonomous system numbers of other systems on your network.
  - Set the BGP statement to yes.
  - Set up a peer clause for each neighbor in that autonomous system. For example:

```
# Perform all BGP operations

 bgp yes {
         peer 192.9.201.1 ;
 } ;
```

- If using SNMP:
  - Set the SNMP statement to yes.

```
snmp yes ;
```

Configuring the gated Daemon To Run IPv6

To configure the **gated** daemon to run under Internet Protocol version 6 (IPv6), first ensure that your system has been configured for IPv6 and IPv6 routing:

1. Run **autoconf6** to automatically configure your interfaces for IPv6.
2. Configure site local addresses for each IPv6 interface on which you want to use IPv6 routing using the following command:

```
ifconfig   interface   inet6 fec0:   n ::   address  /64 alias
```

where

interface Is the name of the interface, such as tr0 or en0.

n Is any decimal number; for example, 11

address Is the portion of the IPv6 interface address that follows the double colons; for example, given the IPv6 address `fe80::204:acff:fe86:298d`, the *address* entry would be `204:acff:fe86:298d`.

Note:: You can use the command **netstat –i** to see what your IPv6 address is for each configured interface.

If token ring `tr0` has an IPv6 address of `fe80::204:acff:fe86:298d`, you issue the following command:

```
ifconfig tr0 inet6 fec0:13::204:acff:fe86:298d/64 alias
```

3. Turn on IPv6 forwarding with the following command:

```
no –o ip6forwarding=1
```

4. Start **ndpd–router** with the following command:

```
ndpd–router –g
```

See **ndpd–router** to determine which flags to use for your network configuration.

Starting **ndpd–router** allows your system to act as a router for the Neighbor Discovery Protocol. Neighbor Discovery Protocol routers inform Neighbor Discovery hosts with routing information so hosts can route IPv6 packets.

Any hosts on the network that you want to be part of the IPv6 network must run **ndpd–host**. Hosts on the network that run **ndpd–host** will recognize themselves as part of an IPv6 network and use Neighbor Discovery Protocol, which allows them to determine and monitor link–layer addresses both to allow neighbor routing and to find neighboring routers for forwarding packets.

See **ndpd–router**, **ndpd–host**, or read RFC 1970, *Neighbor Discovery*, for more information.

Next, configure the **gated** daemon:

1. Decide which IPv6 gateway protocols are most appropriate for your system. The choices for IPv6 routing protocols are Border Gateway Protocol enhanced for IPv6 (BGP4+) and Routing Information Protocol Next Generation (RIPng).
2. Edit the **etc/gated.conf** file to reflect the desired **gated** daemon configuration.

   Note:: AIX 4.3.2 and later run **gated** version 3.5.9. The syntax of the **gated.conf** file has changed slightly from earlier versions. Read the **gated.conf** documentation or use the sample file that is shipped in the **/usr/sample/tcpip** directory for correct syntax.

   When configuring **BGP4+** or **RIPng**, use IPv6 addresses in which the syntax specifies an IP address.

   Note:: By default, **RIPng** multicasts its packets.

Once the **/etc/gated.conf** file has been modified, the **gated** daemon can be started.

Getting an Autonomous System Number

If you use **EGP** or **BGP**, you should obtain an official *autonomous system number* for your gateway. To obtain an official autonomous system number, contact the NIC at INFO@INTERNIC.NET.

Mobile IPv6

Mobile IPv6 provides mobility support in for IPv6. It allows you to keep the same internet address all over the world, and allows applications using that address to maintain transport and upper–layer connections when you change locations. It allows mobility across homogenous and heterogeneous media. For example, Mobile IPv6 facilitates node movement from an Ethernet segment to a wireless LAN cell, with the mobile node's IP address remaining unchanged.

In Mobile IPv6, each mobile node is identified by two IP addresses: its home address and its care–of address. The home address is a permanent IP address that identifies the mobile node regardless of its location. The care–of address changes at each new point of attachment and provides information about the mobile node's current situation. When a mobile node arrives to a visited network, it must acquire a care–of address, which will be used during the time that the mobile node is under this location in the visited network. It may use the methods of IPv6 Neighborhood Discovery to get the care–of address (see Neighbor Discovery/Stateless Address Autoconfiguration on page 4-10). Both stateless and stateful autoconfiguration are possible. The care–of address can also be manually configured. How the care–of address is acquired is irrelevant to Mobile IPv6.

There must be at least one home agent configured on the home network, and the mobile node must be configured to know the IP address of its home agent. The mobile node sends a packet containing a binding update destination option to the home agent. The home agent gets the packet and makes an association between the home address to the mobile node and the care–of address it received. The home agent responds with a packet containing a binding acknowledgment destination option.

The home agent keeps a binding cache containing associations between the home addresses and the care–of addresses for the mobile nodes it serves. The home agent will intercept any packets destined for the home address and forward them to the mobile nodes. A mobile node will then send a binding update to the correspondent node informing it of its care–of address, and the correspondent node will create a binding cache entry so that it can send future traffic directly to the mobile node at its care–of address.

Mobility support in AIX provides the following basic functions:

As a **Home Agent** node:

- Maintain an entry in its binding cache for each mobile node for which it is serving.
- Intercept packets addressed to a mobile node for which it is currently serving as the home agent, on that mobile node's home link, while the mobile node is away from home.
- Encapsulate such intercepted packets in order to tunnel them to the primary care–of address for the mobile node indicated in its binding in the home agent's binding cache.
- Return a binding acknowledgment option in response to a binding update option received with the acknowledge bit set.

As a **Stationary Correspondent** Node:

- Process a home address option received in any IPv6 packet
- Process a binding update option received in a packet and to return a binding acknowledgement option if the acknowledge (A) bit is set in the received binding update
- Maintain a binding cache of the bindings received in accepted binding updates
- Send packets using a routing header when there is a binding cache entry for a mobile node that contains the mobile node's current care–of address

As a **Router** Node in a Network visited by the mobile

- Send an advertisement interval option in its router advertisements to aid movement detection by mobile nodes. It is configurable by the `–m` parameter in the **ndpd–router** daemon
- Support sending unsolicited multicast router advertisements at the faster rate described in RFC 2461. It is configurable by the parameter `–m` in the **ndpd–router** daemon

IP Security

- Tunnels must be statically predefined using the home addresses between the home agent and the mobile node or between the correspondent and the mobile node
- Only AH–ESP in transport mode is supported
- When filtering on protocol 60, only packets with the destination options BU, BA and BR are securized
- When filtering on all the traffic, all Mobility Packets (BU, BA signalization and other packets containing also data) are securized
- IP Security filtering on protocol 60 should always be used when mobility is in use. Some mobile nodes may not accept BA and BU packets unless IP Security is used, and accepting these packets when IP Security was not used can be a serious security problem.

IP Security with IKE

- IKE acts as responder on the home agent or the correspondent
- Only aggressive mode supported

Configure Mobile IPv6

Start Mobile IPv6 with IP Security

Home Agent

1. Define IKE tunnels (phase 1 and phase 2) and responder and for a AH protocol in the database between the home agent IP address and the home address of each mobile node on the home agent is susceptible to communicate with. See IP Security in the AIX Security Guide for details.
2. Define the AH IP Security Association between the home agent IP address and each mobile home address the correspondent is susceptible to communicate with.
3. Run the following command:

```
/etc/rc.mobip6 start –H –S
```

Correspondent

1. Define IKE tunnels (phase 1 and phase 2) and responder and for a AH protocol in the database between the home agent IP address and the home address of each mobile node on the home agent is susceptible to communicate with. See IP Security in the AIX Security Guide for details.
2. Define the AH IP Security Association between the home agent IP address and each mobile home address the correspondent is susceptible to communicate with.
3. Run the following command:

```
/etc/rc.mobip6 start –S
```

Router

Run the following to facilitate movement detection:

```
ndpd–router –m
```

Start Mobile IPv6 without IP Security

Although Mobile IPv6 can be started without IP Security, this is not recommended. IP Security protects the binding packets (the filtering on protocol 60). Using Mobile IPv6 without IP Security leaves a security hole.

Home Agent

Run the following command:

```
/etc/rc.mobip6 start –H
```

Correspondent

Run the following command:

```
/etc/rc.mobip6 start
```

Router

Run the following to facilitate the movement detection:

```
ndpd–router –m
```

Stopping Mobile IPv6

To stop Mobile IPv6 and leave the system functioning as an IPv6 gateway, run the following command:

```
/etc/rc.mobip6 stop
```

To stop mobile IPv6 and disable IPv6 gateway functionality, run the following command:

```
/etc/rc.mobip6 stop –N –F
```

Troubleshooting Mobile IPv6

- Get the binding states by running the following:

```
mobip6ctrl –b
```

- See TCP/IP Problem Determination for information on using the TCP/IP troubleshooting utilities.

Virtual IP Address (VIPA)

A virtual IP address eliminates a host's dependency upon individual network interfaces. Incoming packets are sent to the system's VIPA address, but all packets travel through the real network interfaces.

Previously, if an interface failed, any connections to that interface were lost. With VIPA on your system and routing protocols within the network providing automatic reroute, recovery from failures occurs without disruption to the existing user connections that are using the virtual interface as long packets can arrive through another physical interface. Systems running VIPA are more highly available because adapter outages no longer affect active connections. Since multiple physical adapters carry the system IP traffic, overall load is not concentrated on a single adapter and associated subnet.

The AIX VIPA function is transparent to the network equipment. No special network equipment or other hardware is needed. To implement VIPA , you need to have the following:

- two or more existing IP interfaces of any physical type on different subnets that connect into the corporate network
- IP routing protocols running within the corporate network

Configuring VIPA

VIPA is configured, just as any IP network interface, in SMIT. In addition, you can specify a group of interfaces while configuring VIPA. When configured this way, for all the outgoing connections initiated by the VIPA host via these interfaces, which are designated to use a VIPA, the virtual address becomes the source address placed in the TCP/IP packet header of the outgoing packets.

1. For an IPv4 VIPA, type `smit mkinetvi` on the command line. For an IPv6 VIPA, type `smit mkinetvi6` on the command line.
2. Fill in the reqiured fields, and press Enter.

Managing VIPA

The following topics are covered in this section:

- Add an adapter to a VIPA on page 4-194
- Remove an adapter from a VIPA on page 4-195
- Sample VIPA Environment in AIX 5.2 on page 4-195
- Additional Technical Information on page 4-196

Add an adapter to a VIPA

To add an adapter to your VIPA interface, follow these steps:

1. Type `smit chvi` on the command line.
2. Select the VIPA to which you want to add an adapter and press Enter.
3. Enter the adapter that you want to add in the `Interface Name(s)` field.
4. Enter `ADD` in the `ADD/REMOVE interface(s)` field and press Enter.

Remove an adapter from a VIPA

To remove an adapter from a VIPA, follow these steps:

1. Type `smit chvi` on the command line.
2. Select the VIPA from which you want to remove and adapter, and press Enter.
3. Enter that adapter that you want to remove in the `Interface Name(s)` field.
4. Enter `REMOVE` in the `ADD/REMOVE interface(s)` field and press Enter.

Sample VIPA Environment in AIX 5.2

A system has a virtual IP address, `vi0`, of `10.68.6.1` and two physical connections, `en1` with IP address `10.68.1.1` and `en5`, with IP address `10.68.5.1`. In this example, both physical connections are Ethernet, but any mixture of IP interfaces, such as token–ring or FDDI, would be supported as long as the subnets were ultimately attached to the larger corporate network and were known to the corporate routers.

Running the `lsattr –El vi0` command produces the following results:

```
netaddr          10.68.6.1      N/A
True
 state            up             Standard Ethernet Network Interface
 True
 netmask          255.255.255.0  Maximum IP Packet Size for This Device
 True
 netaddr6                        Maximum IP Packet Size for REMOTE Networks
True
 alias6                          Internet Address
 True
 prefixlen                       Current Interface Status
 True
 alias4                          TRAILER Link–Level Encapsulation
 True
 interface_names  en1,en5        Interfaces using the Virtual Address
 True
```

Running the `ifconfig vi0` command produces the following results:

```
vi0: flags=84000041<UP,RUNNING,64BIT>
        inet 10.68.6.1 netmask 0xffffff00
        iflist : en1 en5
```

Running the `netstat –rn` command produces the following results:

```
Routing tables
 Destination       Gateway       Flags     Refs    Use    If     PMTU Exp
Groups

 Route Tree for Protocol Family 2 (Internet):
 default           10.68.1.2      UG        3      1055   en1     –   –
 10.68.1/24        10.68.1.1      U         0      665    en1     –   –
 10.68.5/24        10.68.5.1      U         0      1216   en5     –   –
 127/8             127.0.0.1      U         4      236    lo0     –   –
 10.68.6.1         127.0.0.1      UH        0      0      lo0     –   –
```

The outgoing packets that do not have a source address set and that are routed via interfaces `en1` and `en5` will have the source address set to the virtual address (`10.68.6.1`). Incoming packets are routed to the VIPA address (`10.68.6.1`) advertised on the network. Because `vi0` is virtual — not associated with any device — there should be no entries for it in the system–wide routing table displayed via the `netstat –rn` command. This means no interface route is added when the interface is configured in SMIT.

If one of the physical interfaces, a network attachment, or a network path fails, the network protocols route to the other physical interface on the same system. If a remote system telnets to the `vi0` address, packets to `vi0` can arrive using either `en1` or `en5`. If `en1` is down, for example, packets can still arrive on `en5`. Note that routing protocols might take time to propagate the routes.

When using the VIPA, the end systems and intervening routers must be able to route the packets destined for VIPA (`vi0`) to one of the physical interfaces (`en1` or `en5`).

Additional Technical Information

VIPA versus alias

The VIPA concept is similar to IP aliases except that the addresses are not associated with a hardware interface. VIPA offers several advantages that IP aliases does not:

- VIPA offers a virtual device that can be brought up and down independently without affecting the physical interfaces
- VIPA addresses can be changed while aliases can only be added or deleted

Access via the IP address of the real adapters

Individual interfaces are still accessible to other systems after VIPA is implemented. However, using the real IP addresses for ping and telnet sessions sidesteps the VIPA advantage of communicating independent of the physical adapters. VIPA hides physical adapter failures from the outlying clients. Using the real addresses reintroduces the dependency upon the physical adapters.

If the remote system contacts the VIPA system using the VIPA address or if an application on the VIPA system initiates the communication to another system, the VIPA address will be used as the source IP address in the packet. However, if the remote system initiates the session using the IP address of the real interface, that real IP address will be the source IP address in the responding packets. There is one exception. For applications that bind to a particular IP interface, the outgoing packets will carry the source address of the interface to which they are bound.

VIPA and routing protocols

The gated daemon was modified for VIPA so that it would not add the interface route or send advertisements over virtual interfaces. The OSPF protocol, supported by gated, will advertise the virtual interface to neighboring routers. The other hosts on the network will be able to talk to the VIPA host through the first–hop router.

Multiple VIPA addresses

Multiple virtual interfaces may be configured.

Multiple VIPA interfaces would be useful, for example, if network routers could give preferential treatment to packets sent to or from certain VIPA addresses. Or, you might use multiple VIPA interfaces if they were binding applications to a specific VIPA interface. For example, to run multiple web servers for multiple companies on a single machine, you could configure the following:

- vi0 200.1.1.1 www.companyA.com
- vi1 200.1.1.2 www.companyB.com
- vi2 200.1.1.3 www.companyC.com

VIPA on AIX 5.1

It was also not possible to specify a group of interfaces that use a particular VIPA in AIX 5.1. The first VIPA in the address list would get chosen as the default source address when the application does not explicitly bind to an address.

EtherChannel and IEEE 802.3ad Link Aggregation

EtherChannel and IEEE 802.3ad Link Aggregation are network port aggregation technologies that allow several Ethernet adapters to be aggregated together to form a single pseudo Ethernet device. For example, `ent0` and `ent1` can be aggregated to `ent3`; interface `en3` would then be configured with an IP address. The system considers these aggregated adapters as one adapter. Therefore, IP is configured over them as over any Ethernet adapter. In addition, all adapters in the EtherChannel or Link Aggregation are given the same hardware (MAC) address, so they are treated by remote systems as if they were one adapter.

The main benefit of EtherChannel and IEEE 802.3ad Link Aggregation is that they have the network bandwidth of all of their adapters in a single network presence. If an adapter fails, the packets are automatically sent on the next available adapter without disruption to existing user connections. The adapter is automatically returned to service on the EtherChannel or Link Aggregation when it recovers.

There are some differences between EtherChannel and IEEE 802.3ad Link Aggregation. Consider the differences given in the following table to determine which would be best for your situation.

EtherChannel	IEEE 802.3ad
Requires configuration of switch	Little, if any, configuration of switch required to form aggregation. Some initial setup of the switch may be required.
Can operate in multiple modes	Only operates in standard mode

For more information on configuring and using EtherChannel, see EtherChannel on page 4-197. For more information on configuring and using IEEE 802.3ad Link Aggregation, see IEEE 802.3ad Link Aggregation on page 4-204. For information on the different AIX and switch configuration combinations and the results they will produce, see Interoperability Scenarios on page 4-207.

EtherChannel

The adapters that belong to an EtherChannel are cabled to the same EtherChannel–enabled network switch, which must be manually configured to identify the ports that belong to the EtherChannel.

Traffic is distributed across the adapters in either the standard way (where the adapter over which the packets are sent is chosen depending on the destination address) or on a round–robin basis (where packets are sent evenly across all adapters). Incoming traffic is distributed in accordance to the switch configuration and is not controlled by the EtherChannel operation mode.

In AIX, users can configure multiple EtherChannels per system, but the standard requires that all the links in one EtherChannel are attached to a single switch. Because the EtherChannel cannot be spread across two switches, the entire EtherChannel is lost if the switch is unplugged or fails. To solve this problem, a new backup option available in AIX 5.2 and later keeps the service running when the main EtherChannel fails. The backup and EtherChannel adapters should be attached to different network switches. In the event that all of the adapters in the EtherChannel fail, the IP and MAC addresses will be automatically moved to the backup adapter. When any link in the EtherChannel is restored, the service is moved back to the EtherChannel.

Network Interface Backup, a mode of operation available for EtherChannel in AIX 4.3.3 and AIX 5.1, protects against a single point of Ethernet network failure. No special hardware is required to use Network Interface Backup, but the backup adapter should be connected a separate switch. In Network Interface Backup mode, only one adapter at a time is actively used for network traffic. The EtherChannel tests the currently–active adapter and, optionally, the network path to a user–specified node. When a failure is detected, the MAC and IP addresses are moved to the next adapter, which will be used until it fails. Network Interface Backup provides detection and failover with no disruption to user connections. Network Interface Backup was originally implemented as a mode in the EtherChannel SMIT menu. In AIX 5.2 and later, the backup adapter provides the equivalent function, so the mode was eliminated from the SMIT menu. To use network interface backup in AIX 5.2 and later, see Configure Network Interface Backup.

Configuring EtherChannel

Follow these steps to configure an EtherChannel.

Considerations

- You can have up to eight Ethernet adapters per EtherChannel.
- You can configure multiple EtherChannels on a single system, but each EtherChannel constitutes an additional Ethernet interface. The **no** command option, **ifsize**, may need to be increased to include not only the Ethernet interfaces for each adapter, but also any EtherChannels that are configured. The default **ifsize** is eight.
- You can use any supported Ethernet adapter in EtherChannel. However, the Ethernet adapters must be connected to a switch that supports EtherChannel. See the documentation that came with your switch to determine if it supports EtherChannel.
- All adapters in the EtherChannel should be configured for the same speed (100 Mbps, for example) and must be full duplex.
- The adapters that you plan to use for your EtherChannel must not have an IP address configured on them before you start this procedure. Use the **ifconfig** command to unconfigure your adapters. For example, `ifconfig en5 detach` would unconfigure adapter `en5`.
- The adapters used in the EtherChannel cannot be accessed by the system after the EtherChannel is configured. Their attributes must be configured before the EtherChannel is created.
- The adapters to be added to the EtherChannel cannot have interfaces configured in the `up` state in ODM (as would happen if their IP addresses were configured using SMIT). This may cause problems bringing up the EtherChannel when the machine is rebooted because the underlying interface is configured before the EtherChannel with the information found in ODM. Therefore, when the EtherChannel is configured, it finds that one of its adapters is already being used. To change this, before creating the EtherChannel, type `smit chinet`, select each of the interfaces of the adapters to be included in the EtherChannel, and change its **state** value to `detach`. This will ensure that when the machine is rebooted the EtherChannel can be configured without errors.
- If you will be using 10/100 Ethernet adapters in the EtherChannel, you need to enable link polling on those adapters before you add them to the EtherChannel. Type `smitty chgenet` at the command line. Change the **Enable Link Polling** value to `yes`, and press Enter. Do this for every 10/100 Ethernet adapter that you will be adding to your EtherChannel. If you don't this, the EtherChannel will operate, but the individual link failure detection and failover aspect will not work.
- If you plan to use jumbo frames, you need to enable this feature in every adapter before creating the EtherChannel and in the EtherChannel itself. Type `smitty chgenet` at the command line. Change the **Enable Jumbo Frames** value to `yes` and press Enter. Do this for every adapter for which you want to enable Jumbo Frames. You will enable jumbo frames in the EtherChannel itself later.

Configure an EtherChannel

1. Type `smit etherchannel` at the command line.
2. Select **Add an EtherChannel / Link Aggregation** from the list and press Enter.
3. Select the primary Ethernet adapters that you want on your EtherChannel and press Enter. If you are planning to use EtherChannel backup, do not select the adapter that you plan to use for the backup at this point. The EtherChannel backup option is available in AIX 5.2 and later.

   Note:: The **Available Network Adapters** displays all Ethernet adapters. If you select an Ethernet adapter that is already being used (has an interface defined), you will get an error message. You first need to detach this interface if you want to use it.

4. Enter the information in the fields according to the following guidelines:

   – **EtherChannel / Link Aggregation Adapters:** You should see all primary adapters that you are using in your EtherChannel. You selected these adapters in the previous step.

   – **Enable Alternate Address:** This field is optional. Setting this to `yes` will enable you to specify a MAC address that you want the EtherChannel to use. If you set this option to `no`, the EtherChannel will use the MAC address of the first adapter.

   – **Alternate Address:** If you set **Enable Alternate Address** to `yes`, specify the MAC address that you want to use here. The address you specify must start with `0x` and be a 12–digit hexadecimal address.

   – **Enable Gigabit Ethernet Jumbo Frames:** This field is optional. In order to use this, your switch must support jumbo frames. This will only work with a Standard Ethernet (en) interface, not an IEEE 802.3 (et) interface. Set this to `yes` if you want to enable it.

   – **Mode:** You can choose from the following modes:

     . **standard**: In this mode the EtherChannel uses the destination IP address to choose which adapter it will send the packets out on. The EtherChannel divides the last byte of the packet’s destination IP address by the number of adapters in the EtherChannel and uses the remainder (using the modulus operator) to identify the outgoing link. For instance, if the destination IP is 10.10.10.1, and there are 2 adapters in the EtherChannel, (1 / 2) = 0 with remainder 1, so the second adapter is used (the adapters are numbered starting from 0). The adapters are numbered in the order they are listed in the SMIT menu. For non–IP traffic (such as ARP), the last byte of the destination MAC address is used to do the calculation. This mode will guarantee packets are sent out over the EtherChannel in the order they were received, but it may not make full use of the bandwidth. This is the default operation mode.

     . **round_robin**: In this mode the EtherChannel will rotate through the adapters, giving each adapter one packet before repeating. The packets may be sent out in a slightly different order than they were given to the EtherChannel, but it will make the best use of its bandwidth.

     . **netif_backup**: This option is available in AIX 5.1 and AIX 4.3.3. In this mode, the EtherChannel will activate only one adapter at a time. The intention is that the adapters are plugged into different Ethernet switches, each of which is capable of getting to any other machine on the subnet or network. When a problem is detected either with the direct connection (or optionally through the inability to ping a machine), the EtherChannel will deactivate the current adapter and activate a backup adapter. This mode is the only one that makes use of the **Internet Address to Ping**, **Number of Retries**, and **Retry Timeout** fields.

       Network Interface Backup Mode does not exist as an explicit mode in AIX 5.2 and later. To enable Network Interface Backup Mode in AIX 5.2 and later, you must configure one adapter in the main EtherChannel and a backup adapter. For more information, see Configure Network Interface Backup on page 4-200.

- **Backup Adapter:** This field is optional. Enter the adapter that you want to use as your EtherChannel backup. EtherChannel backup is available in AIX 5.2 and later.
- **Internet Address to Ping:** This field is optional and is only available if you are running **Network Interface Backup** mode. The EtherChannel will ping the IP address that you specify here. If the EtherChannel is unable to ping this address for the **Number of Retries** times in **Retry Timeout** intervals, the EtherChannel will switch adapters.
- **Number of Retries:** Enter the number of ping response failures that are allowed before the EtherChannel switches adapters. The default is three. This field is optional and valid only if you have set an **Internet Address to Ping**.
- **Retry Timeout:** Enter the number of seconds between the times when the EtherChannel will ping the **Internet Address to Ping**. The default is one second. This field is optional and valid only if you have set an **Internet Address to Ping**.

5. Press Enter after changing the desired fields to create the EtherChannel.
6. Configure IP over the newly created EtherChannel device by typing `smit chinet` at the command line.
7. Select your new EtherChannel interface from the list.
8. Fill in all the required fields and press Enter.

Configure Network Interface Backup

Network Interface Backup protects against a single point of network failure by providing failure detection and failover with no disruption to user connections. When operating in this mode, only one adapter is active at any given time. If the active adapter fails, the next adapter in the EtherChannel will be used for all traffic. When operating in Network Interface Backup mode, it is not necessary to connect to EtherChannel–enabled switches.

The Network Interface Backup setup is most effective when the adapters are connected to different network switches, as this provides greater redundancy than connecting all adapters to one switch. When connecting to different switches, make sure there is a connection between the switches. This provides failover capabilities from one adapter to another by ensuring that there is always a route to the currently–active adapter.

In releases previous to AIX 5.2, Network Interface Backup mode was implemented as an explicit mode of operation in the EtherChannel SMIT menu. In AIX 5.2 and later, however, the backup adapter functionality provides the equivalent behavior, so the mode was eliminated from the SMIT menu.

Additionally, AIX 5.2 and later versions provide priority, meaning that the adapter configured in the primary EtherChannel will be used over the backup adapter. As long as the primary adapter is functional, it will be used. This contrasts from the behavior of Network Interface Backup mode, where the backup adapter was used until it also failed, regardless of whether the primary adapter had already recovered.

While operating in Network Interface Backup Mode, it is also possible to configure the EtherChannel to detect link failure and network unreachability. To do this, specify the IP address of a remote host where connectivity should always be present. The EtherChannel will periodically ping this host to determine whether there is still a network path to it. If a specified number of ping attempts go unanswered, the EtherChannel will fail over to the next backup adapter in the hope that there is a network path to the remote host through the next adapter. In this setup, not only should every adapter be connected to a different switch, but each switch should also have a different route to the host that is pinged.

This ping feature is only available in Network Interface Backup mode. However, in AIX 5.2 and later, if the ping feature is enabled and a failover has occurred, the EtherChannel will not fail back to the primary adapter. The backup adapter will be the active channel as long as it is working because there is no way to know when the route to the pinged host will become reachable from the primary adapter. If a failure is detected while the backup adapter is active (that is, if either the ping attempts fail from the backup adapter or if the backup adapter itself fails), the EtherChannel will then failover to the primary adapter. If the failover occurred because the primary adapter failed, the EtherChannel will then come back to the primary adapter as soon it has come back up.

To configure Network Interface Backup in AIX 5.2, see Configure Network Interface Backup in AIX 5.2 and later. To configure Network Interface Backup in previous versions of AIX, see Appendix B. Configure Network Interface Backup in previous AIX versions

Configure Network Interface Backup in AIX 5.2 and later

1. With root authority, type `smit etherchannel` on the command line.
2. Select **Add an EtherChannel / Link Aggregation** from the list and press Enter.
3. Select the primary Ethernet adapter and press Enter. This is the adapter that will be used until it fails.

   Note:: The **Available Network Adapters** displays all Ethernet adapters. If you select an Ethernet adapter that is already being used, you will get an error message and will need to detach this interface before you can use it. See the **ifconfig** command for information on how to detach an interface.
4. Enter the information in the fields according to the following guidelines:
   - **EtherChannel / Link Aggregation Adapters**: You should see the primary adapter you selected in the previous step.
   - **Enable Alternate Address**: This field is optional. Setting this to `yes` will enable you to specify a MAC address that you want the EtherChannel to use. If you set this option to `no`, the EtherChannel will use the MAC address of the primary adapter.
   - **Alternate Address**: If you set **Enable Alternate Address** to `yes`, specify the MAC address that you want to use here. The address you specify must start with `0x` and be a 12–digit hexadecimal address.
   - **Enable Gigabit Ethernet Jumbo Frames**: This field is optional. In order to use this, your switch must support jumbo frames. This will only work with a Standard Ethernet (en) interface, not an IEEE 802.3 (et) interface. Set this to `yes` if you want to use it.
   - **Mode**: It is irrelevant which mode of operation you select because there is only one adapter in the main EtherChannel. All packets will be sent over that adapter until it fails. There is no `netif_backup` mode because that mode can be emulated using a backup adapter.
   - **Backup Adapter**: Enter the adapter that you want to be your backup adapter. After a failover, this adapter will be used until the primary adapter recovers. It is recommended to use the preferred adapter as the primary adapter.
   - **Internet Address to Ping**: The field is optional. The EtherChannel will ping the IP address that you specify here. If the EtherChannel is unable to ping this address for **Number of Retries** time in **Retry Timeout** intervals, the EtherChannel will switch adapters.
   - **Number of Retries**: Enter the number of ping response failures that are allowed before the EtherChannel switches adapters. The default is three. This field is optional and valid only if you have set an **Internet Address to Ping**.
   - **Retry Timeout**: Enter the number of seconds between the times when the EtherChannel will ping the **Internet Address to Ping**. The default is one second. This field is optional and valid only if you have set an **Internet Address to Ping**.
5. Press Enter after changing the desired fields to create the EtherChannel.
6. Configure IP over the new interface by typing `smit chinet` at the command line.

7. Select your new EtherChannel interface from the list.
8. Fill in all the required fields and press Enter.

For additional tasks that can be performed after the EtherChannel is configured, see Managing EtherChannel and IEEE 802.3ad Link Aggregation.

Managing EtherChannel and IEEE 802.3ad Link Aggregation

This section will tell you how to perform the following tasks:

- List EtherChannels or Link Aggregations on page 4-202
- Change the Alternate Address on page 4-202
- Add, remove, or change adapters in an EtherChannel or Link Aggregation on page 4-202
- Remove an EtherChannel or Link Aggregation on page 4-203
- Configure or remove a backup adapter on an existing EtherChannel or Link Aggregation on page 4-203

List EtherChannels or Link Aggregations

1. On the command line, type `smit etherchannel`.
2. Select **List All EtherChannels / Link Aggregations** and press Enter.

Change the Alternate Address

This enables you to specify a MAC address for your EtherChannel or Link Aggregation.

1. Type `ifconfig` *`interface`* `detach`, where *`interface`* is your EtherChannel's or Link Aggregation's interface.
2. On the command line, type `smit etherchannel`.
3. Select **Change / Show Characteristics of an EtherChannel** and press Enter.
4. If you have multiple EtherChannels, select the EtherChannel for which you want to create an alternate address.
5. Change the value in **Enable Alternate EtherChannel Address** to `yes`.
6. Enter the alternate address in the **Alternate EtherChannel Address** field. The address must start with `0x` and be a 12–digit hexadecimal address.
7. Press Enter to complete the process.

Add, remove, or change adapters in an EtherChannel or Link Aggregation

1. Type `ifconfig` *`interface`* `detach`, where *`interface`* is your EtherChannel's interface.
2. On the command line type, `smit etherchannel`.
3. Select **Change / Show Characteristics of an EtherChannel / Link Aggregation** and press Enter.
4. Select the EtherChannel or Link Aggregation that you want to modify.
5. Select the primary adapters that you want in your EtherChannel or Link Aggregation and press Enter. If you are using a backup adapter, do not select that adapter here.
6. Fill in the necessary fields and press Enter.

Remove an EtherChannel or Link Aggregation

1. Type `ifconfig` *`interface`* `detach`, where *`interface`* is your EtherChannel's interface.
2. On the command line type `smit etherchannel`.
3. Select **Remove an EtherChannel /** and press Enter.
4. Select the EtherChannel that you want to remove and press Enter.

Configure or remove a backup adapter on an existing EtherChannel or Link Aggregation

The following procedure configures or removes a backup adapter on an EtherChannel or Link Aggregation. This option is available only in AIX 5.2 and later.

1. Type `ifconfig` *`interface`* `detach`, where *`interface`* is your EtherChannel's or Link Aggregation's interface.
2. On the command line, type `smit etherchannel`.
3. Select **Change / Show Characteristics of an EtherChannel / Link Aggregation**.
4. Select the EtherChannel or Link Aggregation that you are adding or modifying the backup adapter on.
5. Enter the adapter that you want to use as your backup adapter in the **Backup Adapter** field, or select **NONE** if you wish to stop using the backup adapter.

Troubleshooting EtherChannel

If you are having trouble with your EtherChannel, consider the following:

Tracing EtherChannel

Use **tcpdump** and **iptrace** to troubleshoot the EtherChannel. The trace hook id for the transmission packets is 2FA and for other events is 2FB. You cannot trace receive packets on the EtherChannel as a whole, but you can trace each adapter's receive trace hooks.

Viewing EtherChannel Statistics

Use the **entstat** command to get the aggregate statistics of all the adapters in the EtherChannel. For example, `entstat ent7` will display the aggregate statistics of ent7. Adding the `–d` flag will also display the statistics of each adapter individually. For example, typing `entstat –d ent7` will show you the aggregate statistics of the EtherChannel as well as the statistics of each individual adapter in the EtherChannel.

Note:: In the `General Statistics` section, the number shown in `Adapter Reset Count` is the number of failovers. In EtherChannel backup, coming back to the main EtherChannel from the backup adapter is not counted as a failover. Only failing over from the main channel to the backup is counted.

In the `Number of Adapters` field, the backup adapter is counted in the number displayed.

Improving Slow Failover

If the failover time when you are using network interface backup mode or EtherChannel backup is slow, verify that your switch is not running the Spanning Tree Protocol (STP). When the switch detects a change in its mapping of switch port to MAC address, it runs the spanning tree algorithm to see if there are any loops in the network. Network Interface Backup and EtherChannel backup may cause a change in the port to MAC address mapping.

Switch ports have a forwarding delay counter that determines how soon after initialization each port should begin forwarding or sending packets. For this reason, when the main channel is re–enabled, there is a delay before the connection is re–established, whereas the failover to the backup adapter is faster. Check the forwarding delay counter on your switch and make it as small as possible so that coming back to the main channel occurs as fast as possible.

For the EtherChannel backup function to work correctly, the forwarding delay counter must not be more than 10 seconds, or coming back to the main EtherChannel might not work correctly. Setting the forwarding delay counter to the lowest value allowed by the switch is recommended.

Adapters not Failing Over

If adapter failures are not triggering failovers, check to see if your adapter card needs to have link polling enabled to detect link failure. Some adapters cannot automatically detect their link status. To detect this condition, these adapters must enable a link polling mechanism that starts a timer that periodically verifies the status of the link. Link polling is disabled by default. For EtherChannel to work correctly with these adapters, however, the link polling mechanism must be enabled on each adapter before the EtherChannel is created.

Adapters that have a link polling mechanism have an ODM attribute called **poll_link**, which must be set to `yes` for the link polling to be enabled. Before creating the EtherChannel, use the following command on every adapter to be included in the channel:

```
smit chgenet
```

Change the **Enable Link Polling** value to `yes` and press Enter.

Using Jumbo Frames

For the jumbo frames option to work properly, aside from enabling the **use_jumbo_frame** attribute on the EtherChannel, you must also enable jumbo frames on each adapter before creating the EtherChannel using the following command:

```
smitty chgenet
```

Change the **Enable Jumbo Frames** value to `yes` and press Enter.

Remote Dump

Remote dump is not supported over an EtherChannel.

IEEE 802.3ad Link Aggregation

IEEE 802.3ad is a standard way of doing link aggregation. Conceptually, it works the same as EtherChannel in that several Ethernet adapters are aggregated into a single virtual adapter, providing greater bandwidth and protection against failures. Like EtherChannel, IEEE 802.3ad requires support in the switch.

In IEEE 802.3ad, the protocol automatically tells the switch which ports should be aggregated. When an IEEE 802.3ad aggregation is configured, Link Aggregation Control Protocol Data Units (LACPDUs) are exchanged between the server machine and the switch. This Link Aggregation Control Protocol (LACP) will let the switch know that the adapters configured in the aggregation should be considered as one on the switch without further user intervention.

Although the IEEE 802.3ad specification does not allow the user to choose which adapters are aggregated, the AIX implementation does allow the user to select the adapters.

To be able to aggregate adapters (meaning that the switch will allow them to belong to the same aggregation) they must be of the same line speed (for example, all 100 Mbps or 1 Gbps) and they must all be full duplex. If you attempt to place adapters of different line speeds or different duplex modes, the creation of the aggregation on the AIX system will succeed, but the switch may not aggregate the adapters together. If the switch does not successfully aggregate the adapters together, you may notice a decrease in network performance. For information on how to determine whether an aggregation on a switch has succeeded, see Troubleshooting IEEE 802.3ad on page 4-206.

According to the IEEE 802.3ad specification, packets going to the same IP address are all sent over the same adapter. Thus, when operating in `8023ad` mode, the packets will always be distributed in the standard fashion, never in a round–robin fashion.

The backup adapter feature is available for IEEE 802.3ad Link Aggregations. The backup adapter does not need to be connected to an IEEE 802.3ad–enabled switch, but if it is, the backup adapter will still follow the IEEE 802.3ad LACP.

You can also configure an IEEE 802.3ad Link Aggregation if the switch supports EtherChannel but not IEEE 802.3ad. In that case, you would have to manually configure the ports as an EtherChannel on the switch (just as if a regular EtherChannel had been created). By setting the mode to `8023ad`, the aggregation will work with EtherChannel–enabled as well as IEEE 802.3ad–enabled switches. For more information about interoperability, see Interoperability Scenarios on page 4-207 .

Note:: The steps to enable the use of IEEE 802.3ad varies from switch to switch. You should consult the documentation for your switch to determine what initial steps, if any, must be performed to enable LACP in the switch.

For information in how to configure an IEEE 802.3ad aggregation, see Configuring IEEE 802.3ad Link Aggregation on page 4-205.

Considerations

Consider the following before configuring IEEE 802.3ad Link Aggregation:

- Although the AIX implementation of IEEE 802.3ad will allow the Link Aggregation to contain adapters of different line speeds, you should aggregate only adapters that are set to the same line speed and are set to full duplex. This will help avoid potential problems configuring the Link Aggregation on the switch. Refer to your switch's documentation for more information on what types of aggregations your switch allows.
- If you will be using 10/100 Ethernet adapters in the Link Aggregation, you need to enable link polling on those adapters before you add them to the aggregation. Type `smitty chgenet` at the command line. Change the **Enable Link Polling** value to `yes`, and press Enter. Do this for every 10/100 Ethernet adapter that you will be adding to your Link Aggregation.

Configuring IEEE 802.3ad Link Aggregation

Follow these steps to configure an IEEE 802.3ad Link Aggregation:

1. Type `smit etherchannel` at the command line.
2. Select **Add an EtherChannel / Link Aggregation** from the list and press Enter.
3. Select the primary Ethernet adapters that you want on your Link Aggregation and press Enter. If you are planning to use a backup adapter, do not select the adapter that you plan to use for the backup at this point. The backup adapter option is available in AIX 5.2 and later.

   Note:: The **Available Network Adapters** displays all Ethernet adapters. If you select an Ethernet adapter that is already being used (has an interface defined), you will get an error message. You first need to detach these interfaces if you want to use them.

4. Enter the information in the fields according to the following guidelines:
   - **EtherChannel / Link Aggregation Adapters:** You should see all primary adapters that you are using in your Link Aggregation. You selected these adapters in the previous step.
   - **Enable Alternate Address:** This field is optional. Setting this to `yes` will enable you to specify a MAC address that you want the Link Aggregation to use. If you set this option to `no`, the Link Aggregation will use the MAC address of the first adapter.

- **Alternate Address:** If you set **Enable Alternate Address** to `yes`, specify the MAC address that you want to use here. The address you specify must start with `0x` and be a 12–digit hexadecimal address.
- **Enable Gigabit Ethernet Jumbo Frames:** This field is optional. In order to use this, your switch must support jumbo frames. This will only work with a Standard Ethernet (en) interface, not an IEEE 802.3 (et) interface. Set this to `yes` if you want to enable it.
- **Mode:** Enter `8023ad`.
- **Backup Adapter:** This field is optional. Enter the adapter that you want to use as your backup. The backup adapter option is available in AIX 5.2 and later.
- **Internet Address to Ping:** This field is optional, and only available if you have only one adapter in the main aggregation and a backup adapter. The Link Aggregation will ping the IP address that you specify here. If the Link Aggregation is unable to ping this address for the **Number of Retries** times in **Retry Timeout** intervals, the Link Aggregation will switch adapters.
- **Number of Retries:** Enter the number of ping response failures that are allowed before the Link Aggregation switches adapters. The default is three. This field is optional and valid only if you have set an **Internet Address to Ping**.
- **Retry Timeout:** Enter the number of seconds between the times when the Link Aggregation will ping the **Internet Address to Ping**. The default is one second. This field is optional and valid only if you have set an **Internet Address to Ping**.

5. Press Enter after changing the desired fields to create the Link Aggregation.
6. Configure IP over the newly created Link Aggregation device by typing `smit chinet` at the command line.
7. Select your new Link Aggregation interface from the list.
8. Fill in all the required fields and press Enter.

Managing IEEE 802.3ad

For management tasks that can be performed on an IEEE 802.3ad Link Aggregation after configuration, see Managing EtherChannel and IEEE 802.3ad Link Aggregation.

Troubleshooting IEEE 802.3ad

If you are having trouble with your IEEE 802.3ad Link Aggregation, use the following command to verify the mode of operation of the Link Aggregation:

```
entstat –d   device
```

where *`device`* is the Link Aggregation device.

This will also make a best–effort determination of the status of the progress of LACP based on the LACPDUs received from the switch. The following status values are possible:

- `Inactive`: LACP has not been initiated. This is the status when a Link Aggregation has not yet been configured, either because it has not yet been assigned an IP address or because its interface has been detached.
- `Negotiating`: LACP is in progress, but the switch has not yet aggregated the adapters. If the Link Aggregation remains on this status for longer than one minute, verify that the switch is correctly configured. For instance, you should verify that LACP is enabled on the ports.
- `Aggregated`: LACP has succeeded and the switch has aggregated the adapters together.
- `Failed`: LACP has failed. Some possible causes are that the adapters in the aggregation are set to different line speeds or duplex modes or that they are plugged into different switches. Verify the adapters' configurations.

In addition, some switches allow only contiguous ports to be aggregated and may have a limitation on the number of adapters that can be aggregated. Consult the switch documentation to determine any limitations that the switch may have, then verify the switch configuration.

Note:: The Link Aggregation status is a diagnostic value and does not affect the AIX side of the configuration. This status value was derived using a best–effort attempt. To debug any aggregation problems, it is best to verify the switch's configuration.

Interoperability Scenarios

The following table shows several interoperability scenarios. Consider these scenarios when configuring your EtherChannel or IEEE 802.3ad Link Aggregation. Additional explanation of each scenario is given after the table.

AIX configuration mode	Switch configuration	Result
`8023ad`	IEEE 802.3ad LACP	OK – AIX initiates LACPDUs, which triggers an IEEE 802.3ad Link Aggregation on the switch.
`standard` or `round_robin`	EtherChannel	OK – Results in traditional EtherChannel behavior.
`8023ad`	EtherChannel	OK – Results in traditional EtherChannel behavior. AIX initiates LACPDUs, but the switch ignores them.
`standard` or `round_robin`	IEEE 802.3ad LACP	Undesirable – Switch cannot aggregate. The result is poor performance as AIX moves the MAC address between switch ports

- `8023ad` with IEEE 802.3ad LACP:

  This is the most common IEEE 802.3ad configuration. The switch can be set to passive or active LACP.

- `standard` or `round_robin` with EtherChannel:

  This is the most common EtherChannel configuration.

- `8023ad` with EtherChannel:

  In this case, AIX will send LACPDUs, but they will go unanswered because the switch is operating as an EtherChannel. However, it will work because the switch will still treat those ports as a single link.

  Note:: In this case, the `entstat –d` command will always report the aggregation is in the `Negotiating` state.

- `standard` or `round_robin` with IEEE 802.3ad LACP:

  This setup is invalid. If the switch is using LACP to create an aggregation, the aggregation will never happen because AIX will never reply to LACPDUs. For this to work correctly, `8023ad` should be the mode set on AIX.

Path MTU Discovery

For two hosts communicating across a path of multiple networks, a transmitted packet becomes fragmented if its size is greater than the smallest MTU of any network in the path. Because packet fragmentation can result in reduced network performance, it is desirable to avoid fragmentation by transmitting packets with a size is no greater than the smallest MTU in the network path. This size is called the path MTU.

The operating system supports a path MTU discovery algorithm as described in RFC 1191. Path MTU discovery can be enabled for TCP and UDP applications by modifying the **tcp_pmtu_discover** and **udp_pmtu_discover** options of the **no** command. When enabled for TCP, path MTU discovery will automatically force the size of all packets transmitted by TCP applications to not exceed the path MTU. Since UDP applications themselves determine the size of their transmitted packets, UDP applications must be specifically written to utilize path MTU information by using the **IP_FINDPMTU** socket option, even if the **udp_pmtu_discover no** option is enabled. By default, the **tcp_pmtu_discover** and **udp_pmtu_discover** options are disabled on AIX 4.2.1 through AIX 4.3.1, and enabled on AIX 4.3.2 and later.

When the path MTU has been discovered for a network route, a separate host route is cloned for the path. These cloned host routes, as well as the path MTU value for the route, can be displayed using the **netstat –r** command. Accumulation of cloned routes can be avoided by allowing unused routes to expire and be deleted. Route expiration is controlled by the **route_expire** option of the **no** command. Route expiration is disabled by default on AIX 4.2.1 through AIX 4.3.1, and set to 1 minute on AIX 4.3.2 and later.

Since routes can change dynamically, the path MTU value for a path might also change over time. Decreases in the path MTU value will result in packet fragmentation, so discovered path MTU values are periodically checked for decreases. By default, decreases are checked for every 10 minutes, and this value can be changed by modifying the value of the **pmtu_default_age** option of the **no** command.

Increases in the path MTU value can result in a potential increase in network performance, so discovered path MTU values are periodically checked for increases. By default, increases are checked for every 30 minutes, and this value can be changed by modifying the value of the **pmtu_rediscover_interval** option of the **no** command.

If not all of the routers in the network path support RFC 1191, then it might not be possible to determine an exact path MTU value. In these cases, the **mmtu** command can be used to add or delete path MTU values that are attempted. Notes:

1. Path MTU discovery cannot be used on duplicate routes, including those configured for group routing (see Restricting Route Use on page 4-186).
2. Enabling path MTU discovery sets the value of the **arpqsize** option of the **no** command to a minimum value of 5. This value is not decreased if path MTU discovery is subsequently disabled.

Serial Line Interface Protocol (SLIP)

Configuring SLIP over a Modem

To configure Serial Line Interface Protocol (SLIP) between two systems that communicate through a modem, you can use the Web–based System Manager, **wsm**, or use the following procedure, which alternates between the System Management Interface Tool (SMIT) interface and the command line to complete the configuration. For clarity, the following instructions use the names bronze and gold for the two hosts.

1. Physically connect the modems to bronze and gold.
2. To create a tty on bronze using SMIT:
   a. Enter:

      ```
      smit maktty
      ```

   b. Select **rs232** as the type of tty you wish to create.
   c. Select an available serial port, for example **sa0** (system serial port 1).
   d. Select a port number for this tty from the list.
   e. Set the BAUD rate to the baud rate of your modem.
   f. Set Enable LOGIN to disable.
   g. Exit SMIT.
3. Create a tty on gold.

   Follow the same procedure as you did for bronze (in step 2), except set Enable LOGIN to **enable**.

   The rest of these instructions assume that the tty number on both bronze and gold is tty1.
4. Test the physical connection with ATE.
   a. On bronze, enter:

      ```
      ate
      ```

   b. At the Unconnected Main Menu, select the **Alter** subcommand. Set the Rate to the baud rate of your modem and the Device to tty1.
   c. At the Unconnected Main Menu, select the **Connect** subcommand. When ATE prompts you for a phone number, enter the phone number of gold and press Enter.
   d. At this point, you should receive a login prompt for gold. Login.
   e. Return to the connected screen, logout from gold, press **Ctrl–v** (to get to the ATE CONNECTED MAIN MENU), press **t** to terminate the connection, and press **q** to exit ATE.

      Note:: If you do not receive a login prompt, return to step 1 and verify that your configuration is correct. Do not proceed until you can login to gold. Because the tty configuration for use with ATE is slightly different from the configuration for use with SLIP, you must make the following changes:

   a. On bronze, enter:

      ```
      smit chgtty
      ```

   b. On gold, enter:

      ```
      smit chgtty–pdisable tty1
      ```

Select **tty1**, then select **Change/Show TTY Program**. Set Enable LOGIN to disable, then exit SMIT.

5. Add the following line to the **/usr/lib/uucp/Devices** file on both bronze and gold:

   ```
   Direct tty1 – 9600 direct
   ```

   or replace `9600` with whatever your modem speed is.

6. Create a SLIP network interface on bronze.

   a. Enter:

      ```
      smit mkinet1sl
      ```

   b. For TTY PORT for SLIP Network Interface, select **tty1**.

   c. Specify an INTERNET ADDRESS, for example, 130.130.130.1.

   d. Specify the DESTINATION address (of gold), for example, 130.130.130.2.

   e. Specify the BAUD RATE of your modem.

   f. Specify the DIAL STRING, for example:

      - ”” AT OK ATDT555–1234 CONNECT ””
      - The meaning of this command is: Use `tty1` at `9600` baud. Send AT to the modem. The modem should respond with OK. Dial the phone number `555–1234`. The modem should respond with CONNECT. The spaces before and after the ”” characters are necessary.

   g. Exit SMIT.

7. Create a SLIP network interface on gold.

   Follow the same procedure as you did for bronze (in step 5), except exchange the INTERNET ADDRESS and the DESTINATION address.

8. Add the following two entries to the **/etc/hosts** file on both bronze and gold:

   ```
   130.130.130.1   bronze
    130.130.130.2   gold
   ```

   The name you assign must be unique. In other words, if the Token–Ring interface on bronze is already assigned the name `bronze`, assign the SLIP interface a name such as `bronze_slip`.

   Note:: For a simplified interface to the **slattach** command, you might use the script `/usr/sbin/slipcall.`

9. Test the SLIP connection.

   a. On bronze, enter:

      ```
      ping gold
      ```

   b. On gold, enter:

      ```
      ping bronze
       If both tests succeed, the SLIP connection is ready for use. If not,
      return to step 5 and verify that the configuration on both bronze and
      gold is correct.
      ```

Configuring SLIP over a Null Modem Cable

To configure SLIP between two systems that are attached using a null modem cable, you can use the Web–based System Manager, **wsm**, or use the following procedure, which alternates between the System Management Interface Tool (SMIT) interface and the command line to complete the configuration. For clarity, these instructions use the names bronze and gold for the two hosts.

1. Physically connect bronze and gold by the null modem cable. The following cables are required.. (The cables are listed in the order they will be connected from bronze to gold.)

   a. Cable B (part number 00G0943). Serial Port Jumper Cable; two are provided with each system, except models 220, 340, and 350 do not require them.

   b. Cable D (part number 6323741, feature code 2936). Asynchronous Cable EIA–232/V.24.

   c. Cable E (part number 59F2861, feature code 2937). Printer/Terminal Interposer EIA–232 (null modem cable).

   d. Changer Adapter (both sides of the adapter are sockets).

2. Create a tty on bronze.

   a. Enter:

   ```
   smit maktty
   ```

   b. Select **rs232** as the type of tty you wish to create.

   c. Select an available serial port, for example **sa0** (system serial port 1).

   d. Select a port number for this tty from the list.

   e. Set the BAUD rate to 19200. (Later, you will change this to 38400. But for now, use 19200.)

   f. Set Enable LOGIN to disable, and then exit SMIT.

3. Create a tty on gold.

   Follow the same steps as you did for bronze (in step 2), except set Enable LOGIN to **enable**.

   Note:: The rest of these instructions assume that the tty number on both bronze and gold is tty1.

4. Test the physical connection with ATE.

   a. On bronze, enter:

   ```
   ate
   ```

   b. At the Unconnected Main Menu, select the **Alter** subcommand. Set the Rate to 19200, and the Device to tty1.

   c. At the Unconnected Main Menu, select the **Connect** subcommand. When ATE prompts you for a phone number, press Enter. You should receive the message:

   ```
   ate: 0828–010 The Connect command has made a connection through port
   tty1
   ```

   d. Press Enter. You should receive a login prompt for gold. Login to gold.

   e. Finally, return to the connected screen, logout from gold, press **Ctrl–v** (to get to the ATE CONNECTED MAIN MENU), press **t** to terminate (end) the connection, and press **q** to exit ATE.

Note:: If you do not receive a login prompt, return to step 1 and verify that your configuration is correct. Do not proceed until you can login to gold. Since the tty configuration for use with ATE is slightly different from the configuration for use with SLIP, you must make the following changes:

a. On bronze, enter:

```
smit chgtty
```

Select **tty1**. Set the BAUD rate to 38400, and then exit SMIT.

b. On gold, enter:

```
pdisable tty1
```

c. On gold, enter:

```
smit chgtty
```

Select **tty1**. Set Enable LOGIN to disable, set the BAUD rate to 38400, and then exit SMIT.

5. Add the following line to the **/usr/lib/uucp/Devices** file on both bronze and gold:

```
Direct tty1 – 38400 direct
```

6. Create a SLIP network interface on **bronze**.

a. Enter:

```
smit mkinet1sl
```

b. For TTY PORT for SLIP Network Interface, select **tty1**.

c. Specify an INTERNET ADDRESS, for example 130.130.130.1.

d. Specify the DESTINATION address (of gold), for example, 130.130.130.2, and then select OK or Enter.

7. Create a SLIP network interface on gold.

Follow the same procedure as you did for bronze (in step 5), except exchange the INTERNET ADDRESS and the DESTINATION address.

8. Add the following two entries to the **/etc/hosts** file on both bronze and gold:

```
130.130.130.1   bronze
 130.130.130.2   gold
```

The name you assign must be unique. In other words, if the Token–Ring interface on bronze is already assigned the name `bronze`, assign the SLIP interface a name such as `bronze_slip`.

9. Start SLIP on both bronze and gold.

Enter:

```
slattach tty1
```

10. Test the SLIP connection.

a. On bronze, enter:

```
ping gold
```

b. On gold, enter:

```
ping bronze
 If both tests succeed, the SLIP connection is ready for use. If not,
return to step 5 and verify that the configuration on both bronze and
gold is correct.
```

Deactivating a SLIP Connection

To deactivate a SLIP connection:

1. Enter:

   ```
   ps –ef | grep slatt
   ```

   Note the process numbers of processes associated with the **slattach** command.

2. For each process number, enter:

   ```
   kill   process_number
   ```

   Do not use the **–9** flag of the **kill** command.

   If **slattach** is accidentally killed with a **–9** flag, a slip lock might remain in /etc/locks. Delete this lock file to clean up after **slattach**.

Removing a TTY

To remove a tty, you can use the Web–based System Manager, **wsm**, or the System Management Interface Tool (SMIT) fast path, **smit rminet**.

Asynchronous Point–to–Point Protocol (PPP) Subsystem

The Asynchronous Point–to–Point Protocol (PPP) subsystem provides an alternative to SLIP. PPP provides a standard method for transporting multiprotocol datagrams over point–to–point media. PPP is comprised of three main layers:

1. A method for encapsulating multiprotocol datagrams. PPP supports the TCP/IP network layer protocols.
2. A Link Control Protocol (LCP) for establishing, configuring, and testing the data–link connection. PPP implements this through streams kernel extensions.
3. A family of Network Control Protocols (NCPs) for establishing and configuring different network layer protocols. PPP supports Internet Protocol Control Protocol (IPCP) for negotiating a TCP/IP connection.

This implementation of PPP supports the following Request for Comments (RFCs):

- RFC 1661, *The Point–to–Point Protocol, LCP*
- RFC 1332, *The PPP Internet Protocol Control Protocol (IPCP)*
- RFC 1662, *PPP in HDLC–like Framing*
- RFC 1334, *PPP Authentication Protocols*
- RFC 1990, *PPP Multilink*

PPP differentiates between client and server. This operating system can act as both a client and a server. The distinction is made to simplify configuration. PPP servers tend to allocate a pool of IP addresses among the connections that are being made. There is some correlation between the media devices. This implementation of PPP breaks this correlation. All server PPP connections are allocated on a first–available basis. This facilitates the separation of PPP from the media. The attachment process must request to be linked to the proper type of link.

User–Level Processes

The Asynchronous Point–to–Point Protocol on this operating system utilizes three user–level processes:

1. A control daemon (**pppcontrold**) run by root under the System Resource Controller (**startsrc –s pppcontrold**). The control daemon's function encompasses loading and configuring all kernel extensions associated with the subsystem. It remains running as long as PPP function is required by the operating system.
2. An attachment process (**pppattachd**) that binds a TTY stream to an instance of the Link Control Protocol, Network Control Protocol, and a datagram protocol. An instance of **pppattachd** exists for each active PPP connection in the system. Any user of the attachment process must belong to the **uucp** group and contain **/usr/sbin** within their **PATH** environment variable.
3. A dialer process (**pppdial**) that establishes an outgoing connection. The dialer is intended to be executed by **pppattachd** as the connector program. Its purpose is to interact over the asynchronous device prior to PPP negotiation. This interaction is defined similarly to the UUCP chat dialog format. The dialer capability is provided to assist in establishing a connection with a remote system. The actual session establishment is out of the scope of PPP.

Configuring the Asynchronous Point–to–Point Protocol

You can use Web-based System Manager or SMIT to configure the Asynchronous Point–to–Point Protocol. The following table shows all tasks that you may need when configuring your system. You must have root privileges to perform the tasks in this table.

At a minimum, when you initially configure your system, you must choose the following tasks from the table:

- Add a Link Configuration
- Add a Server Interface (if you are setting up the machine as a PPP server)
- Add a Demand Interface (if you want the machine to support demand connections)
- Manipulate PAP or CHAP Users/Passwords (if you want the machine to support PPP authentication)
- Start PPP to effect your changes (or Stop then Start PPP, if PPP is currently running)

Configuring the asynchronous PPP tasks

Task	**SMIT fast path**	Web-based System Manager Management Environment
Create Link Control Configuration	**smit ppplcp**	Software —> **Network** —> **TCPIP (IPv4 and IPv6)** —> **Point–to–Point (PPP)** —> **Configure the Point–to–Point Link**.
Add a Link Configuration	**smit addlcp**	
Change/Show a Link Configuration	**smit chglcp**	
Remove a Link Configuration [1]	**smit rmlcp**	Software —> **Network** —> **TCPIP (IPv4 and IPv6)** —> **Point–to–Point (PPP)** —> **Configure the Point–to–Point Link** —> **Link Configuration** —> **Remove Link Configuration**.
Create PPP IP Interfaces	**smit pppip**	
Add a Server Interface	**smit addpppserver**	Software —> **Network** —> **TCPIP (IPv4 and IPv6)** —> **Point–to Point (PPP)** —> **Configure the Point–to–Point Link** —> **Server Interfaces** —> **Add/Change Interface**.
Change/Show a Server Interface	**smit listserver**	Software —> **Network** —> **TCPIP (IPv4 and IPv6)** —> **Point–to–Point (PPP)** —> **Configure the Point–to–Point Link** —> **Server Interfaces** —> **Add/Change Interface**.

Remove a Server Interface [1]	**smit rmlistserver**	Software —> **Network** —> **TCPIP (IPv4 and IPv6)** —> **Point–to–Point (PPP)** —> **Configure the Point–to–Point Link** —> **Server Interfaces** —> **Delete Interface**.
Add a Demand Interface	**smit addpppdemand**	Software —> **Network** —> **TCPIP (IPv4 and IPv6)** —> **Point–to–Point (PPP)** —> **Configure the Point–to–Point Link** —> **Demand Interfaces** —> **Add/Change Interface**.
Change/Show a Demand Interface	**smit listdemand**	Software —> **Network** —> **TCPIP (IPv4 and IPv6)** —> **Point–to–Point (PPP)** —> **Configure the Point–to–Point Link** —> **Demand Interfaces** —> **Add/Change Interface**.
Remove a Demand Interface [1]	**smit rmlistdemand**	Software —> **Network** —> **TCPIP (IPv4 and IPv6)** —> **Point–to–Point (PPP)** —> **Configure the Point–to–Point Link** —> **Demand Interfaces** —> **Delete Interface**.
Manipulate PAP users/passwords	**smit ppppap**	
Add a PAP User	**smit addpapuser**	
Change/Show a PAP User	**smit listpapuser**	
Remove a PAP User	**smit rmpapuser**	
Manipulate CHAP users/passwords	**smit pppchap**	
Add a CHAP User	**smit addchapuser**	
Change/Show a CHAP User	**smit listchapuser**	
Remove a CHAP User	**smit rmchapuser**	
Start PPP [2]	**smit startppp**	Software —> **Network** —> **TCPIP (IPv4 and IPv6)** —> **Point–to–Point (PPP)** —> **Start the PPP Subsystem**.
Stop PPP [3]	**smit stopppp**	Software —> **Network** —> **TCPIP (IPv4 and IPv6)** —> **Point–to–Point (PPP)** —> **Stop the PPP Subsystem**.

Notes:

1. Selecting this task destroys the existing information.
2. An alternative way to start PPP is to issue the **startsrc –s pppcontrold** command. However, the SMIT interface also allows you to set PPP to start at boot time.
3. An alternative way to stop PPP is to issue the **stopsrc –s pppcontrold** command. However, the SMIT interface also allows you to have PPP not start at boot time.

PPP and SNMP

PPP can interact with the TCP/IP SNMP daemon to report PPP link layer configuration information as well as information about active Link Control Protocol (LCP) interfaces. Providing that both the TCP/IP SNMP and the SNMP management software are configured correctly, PPP SNMP enables:

- retrieval of PPP Link Configuration information (Maximum Receive Unit size, Asynchronous Character Mapping, etc.)
- setting of PPP Link Configuration information
- retrieval of LCP interface information for active LCP links
- changing of the state of active LCP links can be changed to ”down” by setting the appropriate **ifAdminStatus** Management Information Base (MIB) object

Not all objects defined by RFC1471 for the PPP MIB are supported. Only the **pppLink** table applies to the PPP subsystem, thus the **pppLqr** and **pppTests** portions are not supported. The **pppLink** portion is supported with the following exceptions:

- The **pppLinkConfigMagicNumber** object is read only. In PPP, magic number negotiation is always performed and cannot be disabled.
- The **pppLinkConfigFcsSize** object is read only. PPP only supports FCS sizes of 16 with this operating system.

Enabling PPP SNMP

By default, SNMP for PPP is disabled. To enable PPP SNMP, you can use the Web–based System Manager, **wsm**, or use the following procedure. You must have root privileges to perform this procedure.

Note:: The following procedure assumes that PPP Link Configuration is already set. If not, perform the procedure described in Configuring the Asynchronous Point–to–Point Protocol on page 4-215 before enabling PPP SNMP.

1. Start the SMIT Interface and display the Change/Show a Link Configuration screen by entering:

   ```
   smit chglcp
   ```

2. Toggle the Enable PPP SNMP subagent field to yes.
3. Accept your changes and exit SMIT.

PPP SNMP is not enabled until PPP is restarted.

- If PPP is currently running,
  1. Stop PPP using the **smit stopppp** fast path (see the table in Configuring the Asynchronous Point–to–Point Protocol on page 4-215).
  2. Periodically check to see if the subsystem has completed shutdown by entering:

     ```
     lssrc –s pppcontrold
     ```

     The amount of time it takes to completely stop the subsystem is dependent on the number of links defined in the PPP configuration. The subsystem is completely shut down when the output of this command shows a status of `inoperative`.
  3. Start PPP using the **smit startppp** fast path (see the table in Configuring the Asynchronous Point–to–Point Protocol on page 4-215).
- If PPP is not currently running, start PPP using the **smit startppp** fast path (see the table in Configuring the Asynchronous Point–to–Point Protocol on page 4-215).

TCP/IP Quality of Service (QoS)

Quality of Service (QoS) is a family of evolving Internet standards that provides ways to give preferential treatment to certain types of IP traffic. With the proper support for QoS along a route, this can ameliorate the effects of variable queueing delays and congestion that contribute to poor network performance. The operating system provides host support for QoS to classify outbound traffic into distinct classes of service and to announce and establish resource reservations as requested by client applications.

QoS can be used by an institution to deploy and enforce network policies governing the use of network bandwidth. With QoS, a host can:

- Regulate the amount of traffic of a certain type injected into the network;
- Mark selected packets according to some policy so that subsequent routers can deliver the indicated service;
- Support services such as the virtual leased line service with proper QoS support along the route; and
- Participate in the resource reservation requests from receivers and announce sender sessions available for resource reservation requests.

The QoS support provides the following functions:

- Differentiated services as defined in RFC 2474
- Traffic policing
- In–profile and out–of–profile packet marking
- Traffic shaping
- Metering
- Integrated services for client and server applications as defined in RFC 1633
- RSVP signaling (RFC 2205)
- Guaranteed service (RFC 2212)
- Controlled–Load service (RFC 2211)
- Policy–based networking
- RAPI shared library for application

The QoS subsystem consists of four components:

QoS kernel extension (/usr/lib/drivers/qos)
The QoS kernel extension resides in **/usr/lib/drivers/qos** and is loaded and unloaded using the **cfgqos** and **ucfgqos** configuration methods. This kernel extension enables QoS support.

Policy agent (/usr/sbin/policyd)
The policy agent is a user–level daemon that resides in **/usr/sbin/policyd**. It provides support for policy management and interfaces with the QoS kernel extension to install, modify, and delete policy rules. Policy rules can be defined in the local configuration file (**/etc/policyd.conf**), retrieved from a central network policy server using LDAP, or both.

RSVP agent (/usr/sbin/rsvpd)
The RSVP agent is a user–level daemon that resides in **/usr/sbin/rsvpd**. It implements the RSVP signaling protocol semantics.

RAPI shared library (/usr/lib/librapi.a)

Applications can use the RSVP API (RAPI) to request enhanced quality of service as defined by the Integrated Services Internet QoS model. This library interacts with the local RSVP agent to propagate the QoS request along the path of the data flow using the RSVP protocol. This API is an open standard.

Note:: This implementation of QoS is based on a set of evolving Internet standards and draft standards currently under development by the Internet Engineering Task Force (IETF) and its various working groups. This technology will become more consistent and well defined as these standardization efforts progress within the IETF. It is also important to note that QoS is an emerging Internet technology that is just beginning to be deployed within the Internet. There are many benefits of QoS at all stages of deployment. However, true end–to–end services can only be realized when QoS support exists all along a particular route.

QoS Models

The QoS models for the Internet are open standards defined by the IETF. There are two Internet QoS models currently being standardized within the IETF: *integrated services* and *differentiated services*. These two Internet QoS models augment the traditional best–effort service model described in RFC 1812.

Integrated Services

Integrated Services (IS) is a dynamic resource reservation model for the Internet described in RFC 1633. Hosts use a signaling protocol called Resource ReSerVation Protocol (RSVP) to dynamically request a specific quality of service from the network. QoS parameters are carried in these RSVP messages and each network node along the path installs the parameters to obtain the requested quality of service. These QoS parameters describe one of two currently defined services, guaranteed service and controlled–load service. An important characteristic of IS is that this signaling is done for each traffic flow and reservations are installed at each hop along the route. Although this model is well–suited for meeting the dynamically changing needs of applications, there exist some significant scaling issues that imply it cannot be deployed in a network in which single routers handle many simultaneous flows.

Differentiated Services

Differentiated Services (DS) removes the per–flow and per–hop scalability issues, replacing them with a simplified mechanism of classifying packets. Rather than a dynamic signaling approach, DS uses bits in the IP type of service (TOS) byte to separate packets into classes. The particular bit pattern in the IP TOS byte is called the DS codepoint and is used by routers to define the quality of service delivered at that particular hop, in much the same way routers do IP forwarding using routing table lookups. The treatment given to a packet with a particular DS codepoint is called a per–hop behavior (PHB) and is administered independently at each network node. When the effects of these individual, independent PHBs are concatenated, this results in an end–to–end service.

Differentiated services is being standardized by an IETF working group, which has defined three PHBs: the Expedited Forwarding (EF) PHB, the Assured Forwarding (AF) PHB group, and the Default (DE) PHB. The EF PHB can be used to implement a low latency, low jitter, low loss, end–to–end service such as a virtual leased line (VLL). AF is a family of PHBs, called a PHB group, that is used to classify packets into various drop precedence levels. The drop precedence assigned to a packet determines the relative importance of a packet within the AF class. It can be used to implement the so–called *Olympic* service, which consists of three classes: bronze, silver, and gold. The DE PHB is the traditional best–effort service model as standardized in RFC 1812.

Supported Standards and Draft Standards

The following RFCs and Internet drafts describe the standards on which this QoS implementation is based.

RFC 2474	Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers
RFC 2475	An Architecture for Differentiated Services
RFC 1633	Integrated Services in the Internet Architecture: an Overview
RFC 2205	Resource ReSerVation Protocol (RSVP)
RFC 2210	The Use of RSVP with IETF Integrated Services
RFC 2211	Specification of the Controlled–Load Network Element Service
RFC 2212	Specification of Guaranteed Quality of Service
RFC 2215	General Characterization Parameters for Integrated Service Network Elements
draft–ietf–diffserv–framework–01.txt, October 1998	A Framework for Differentiated Services
draft–ietf–diffserv–rsvp–01.txt, November 1998	A Framework for Use of RSVP with DIFF–serv Networks
draft–ietf–diffserv–phb–ef–01.txt	An Expedited Forwarding PHB
draft–ietf–diffserv–af–04.txt	Assured Forwarding PHB Group
draft–rajan–policy–qosschema–00.txt, October 1998	Schema for Differentiated Services and Integrated Services in Networks
draft–ietf–rap–framework–01.txt, November 1998	A Framework for Policy–based Admission Control[25]
draft–ietf–rap–rsvp–ext–01.txt, November 1998	RSVP Extensions for Policy Control

Note:: QoS is an emerging Internet technology. There are many benefits of QoS at all stages of deployment. However, true end–to–end services can only be realized when QoS support exists all along a particular route.

QoS Installation

QoS is packaged with **bos.net.tcp.server**. This fileset must be installed in order to use QoS. To use the **RAPI** shared library, **bos.adt.include** must also be installed.

QoS Configuration

Stopping and Starting the QoS Subsystem

QoS can be started or stopped through SMIT with the **smit qos** fast path or with the **mkqos** and **rmqos** commands.

To disable the QoS subsystem now and on the next system restart:

```
/usr/sbin/rmqos –B
```

To enable the QoS subsystem now only:

```
/usr/sbin/mkqos –N
```

See the command descriptions for **mkqos** and **rmqos** for the startup and removal command flags.

The **policyd** and **rsvpd** daemons are configured through the **/etc/policyd.conf** and **/etc/rsvpd.conf** configuration files, respectively. These configuration files *must* be edited to customize the QoS subsystem to the local environment. QoS does not work correctly with the supplied example configurations.

Configuring the RSVP agent

The RSVP agent is required if the host is to support the RSVP protocol. The **/etc/rsvpd.conf** configuration file is used to configure the RSVP agent. The syntax of the configuration file is described in the sample configuration file installed in **/etc/rsvpd.conf**.

Example Configuration

```
interface 1.2.3.1
 interface 1.2.3.2 disabled
 interface 1.2.3.3 disabled
 interface 1.2.3.4
 {
   trafficControl
 }

 rsvp 1.2.3.1
 {
   maxFlows 64
 }

 rsvp 1.2.3.4
 {
   maxFlows 100
 }
```

The example illustrates a possible RSVP configuration in which the host has 4 interfaces (virtual or physical) given by the 4 IP addresses, `1.2.3.1, 1.2.3.2, 1.2.3.3,` and `1.2.3.4`.

Interface `1.2.3.1` has been enabled for RSVP. However, traffic control has not been specified and incoming RSVP RESV messages do not cause resource reservation within the TCP subsystem. This interface can support a maximum of 64 simultaneous RSVP sessions.

Interfaces `1.2.3.2` and `1.2.3.3` have been disabled. The RSVP agent cannot use this interface to transmit or receive RSVP messages.

Interface `1.2.3.4` has been enabled for RSVP. In addition, it can install resource reservations into the TCP subsystem in response to an RSVP RESV message. This interface can support up to 100 RSVP sessions.

Any other interfaces present on the host but not mentioned explictly in **/etc/rsvpd.conf** are disabled.

Configuring the Policy Agent

The policy agent is a required component of the QoS subsystem. The **/etc/policyd.conf** configuration file is used to configure the policy agent. The syntax of this configuration file is described in the sample configuration file installed in **/etc/policyd.conf**.

The policy agent can be configured by editing /etc/policyd.conf. Additionally, the following commands are provided to assist in configuring policies:

- **qosadd**
- **qosmod**
- **qoslist**
- **qosremove**

Example Configurations

In the following example, a premium service category is created and used in the tcptraffic policy rule. This service category has a maximum rate of 110000 Kbps, a token bucket depth of 10000 bits, and an outgoing IP TOS value of 11100000 in binary. The tcptraffic policy rule gives this premimum service to all traffic with source IP address given by `1.2.3.6`, destination address `1.2.3.3`, and destination port in the range 0 to 1024.

```
ServiceCategories  premium
 {
    PolicyScope     DataTraffic
    MaxRate         110000
    MaxTokenBucket  10000
    OutgoingTOS     11100000
 }

 ServicePolicyRules   tcptraffic
 {
    PolicyScope    DataTraffic
    ProtocolNumber 6  # tcp
    SourceAddressRange      1.2.3.6–1.2.3.6
    DestinationAddressRange 1.2.3.3–1.2.3.3
    DestinationPortRange    0–1024
    ServiceReference        premium
 }
```

The following statements set up a default service category and use it to restrict the UDP traffic flowing from interfaces `1.2.3.1` through `1.2.3.4` to IP addresses `1.2.3.6` through `1.2.3.10`, port 8000.

```
ServiceCategories  default
 {
    MaxRate         110000
    MaxTokenBucket  10000
    OutgoingTOS     00000000
 }

 ServicePolicyRules   udptraffic
 {
    ProtocolNumber 17  # udp
    SourceAddressRange      1.2.3.1–1.2.3.4
    DestinationAddressRange 1.2.3.6–1.2.3.10
    DestinationPortRange    8000–8000
    ServiceReference        default
 }
```

The followoing example configuration can be used to download rules from an LDAP server using the distinguished subtree name, to lookup the policies on the LDAP server host.

```
ReadFromDirectory
 {
   LDAP_Server      1.2.3.27
   Base             ou=NetworkPolicies,o=myhost.mydomain.com,c=us
 }
```

QoS Problem Determination

The **qosstat** command may be used to display status information about the installed and active policies in the QoS subsystem. This information may be useful to you in determining where a problem exists if you are troubleshooting your QoS configuration. **qosstat** can be used to generate the following report.

```
Action:
   Token bucket rate (B/sec): 10240
   Token bucket depth (B): 1024
   Peak rate (B/sec): 10240
   Min policied unit (B): 20
   Max packet size (B): 1452
   Type: IS–CL
   Flags: 0x00001001 (POLICE,SHAPE)

   Statistics:
     Compliant packets: 1423 (440538 bytes)

   Conditions:
     Source address        Dest address          Protocol
     192.168.127.39:8000   192.168.256.29:35049  tcp       (1 connection)

 Action:
   Token bucket rate (B/sec): 10240
   Token bucket depth (B): 1024
   Peak rate (B/sec): 10240
   Outgoing TOS (compliant): 0xc0
   Outgoing TOS (non–compliant): 0x00
   Flags: 0x00001011 (POLICE,MARK)
   Type: DS

   Statistics:
     Compliant packets: 335172 (20721355 bytes)
     Non–compliant packets: 5629 (187719 bytes)

   Conditions:
     Source address        Dest address          Protocol
     192.168.127.39:80     *:*                   tcp       (1 connection)
     192.168.127.40:80     *:*                   tcp       (5 connections)
```

Policy Specification

This section describes the object classes and attributes used by the policy agent to specify policies for quality of service (QoS) on outgoing traffic. The object classes and attributes are defined, followed by guidelines to enable marking, policing, and shaping.

These conventions are used in the explanations that follow

.

```
      p  : choose one in the allowed parameter set
       B  : integer value of a byte (i.e., 0 =< B =< 255)
       b  : bit string starting with left most bit (e.g., 101 is
            equivalent 10100000 in a byte field)
       i  : integer value
       s  : a character string
       a  : IP address format B.B.B.B
      (R) : Required parameter
      (O) : Optional parameter
```

ReadFromDirectory

This statement specifies parameters for establishing an LDAP session. The `ReadFromDirectory` statement is used in the **/etc/policyd.conf** file to establish the LDAP session.

```
         ReadFromDirectory
          {
            LDAP_Server    a   # IP address of directory server running
LDAP
            LDAP_Port     i    # Port number LDAP server is listening to
            Base     s         # Distinguished Name for LDAP usage
            LDAP_SelectedTag s # Tag to match SelectorTag in object classes
          }
```

where

```
      LDAP_Server (R): IP address of LDAP server
       LDAP_Port   (0): Unique port number, default port is 389
       Base        (R): Example is o=ibm, c=us where o is your organization
and c is country
       LDAP_SelectedTag (R): Unique string matching SelectorTag attribute
in the object class
```

ServiceCategories

This statement specifies the type of service that a flow of IP packets (for example, from a TCP connection or UDP data) should receive end–to–end as they traverse the network. `ServiceCategories` can be repeated with each having a different name so that they can be referred to later. A `ServiceCategories` object requires `ServicePolicyRules` to complete the policy definition.

```
         ServiceCategories   s
          {
             SelectorTag    s     # Required tag for LDAP Search
             MaxRate        i     # Target rate for traffic in this service
class
             MaxTokenBucket i     # The bucket depth
             OutgoingTOS    b     # TOS value of outbound traffic for this
service class
             FlowServiceType   p  # Type of traffic

          }
```

where

```
         s           (R)     : is the name of this service category
          SelectorTag (R)     : Required only for LDAP to Search object
classes
          MaxRate     (O)     : in Kbps (K bits per second), default is 0
          MaxTokenBucket(O)   : in Kb, default is system defined maximum
          OutgoingTOS (O)     : default is 0
          FlowServiceType  (O): ControlledLoad | Guaranteed, default is
ControlledLoad
```

ServicePolicyRules

This statement specifies characteristics of IP packets that are used to match to a corresponding service category. In other words, it defines a set of IP datagrams that should receive a particular service. `ServicePolicyRules` are associated with `ServiceCategories` through the `ServiceReference` attribute. If two rules refer to the same `ServiceCategory`, each rule is associated with a unique instance of the `ServiceCategory`.

```
         ServicePolicyRules   s
          {
             SelectorTag       s   # Required tag for LDAP Search
             ProtocolNumber    i   # Transport protocol id for the policy
rule
             SourceAddressRange      a1–a2
             DestinationAddressRange a1–a2
             SourcePortRange         i1–i2
             DestinationPortRange    i1–i2
             PolicyRulePriority  i       # Highest value is enforced first
             ServiceReference        s  # Service category name which for
this policy rule
          }
```

where

```
         s           (R): is the name of this policy rule
          SelectorTag (R): required only for LDAP to Search object class
          ProtocolNumber     (R): default is 0 which causes no match, must
explicity specify
          SourceAddressRange (O): from a1 to a2 where a2 >= a1, default is
0, any source address
          SourcePortRange    (O): from i1 to i2 where i2 >= i1, default is
0, any source port
          DestinationAddressRange (O): same as SourceAddressRange
          DestinationPortRange    (O): same as SourcePortRange
          PolicyRulePriority (O): Important to specify when ovelapping
policies exist
          ServiceReference   (R): service category this rule uses
```

Guidelines for DiffServ Environments

The following are guidelines to specify policies for marking, shaping, and/or policing in a DiffServ environment.

1. **Marking Only**

```
    OutgoingTOS     : Desired Type Of Service
     FlowServiceType : ControlledLoad
     MaxRate         : Take default of 0
```

2. **Shaping Only**

```
    OutgoingTOS     : Take default of 0
     FlowServiceType : Guaranteed
     MaxRate         : Target rate desired for traffic as a positive
integer
```

3. **Marking and Policing (See Note)**

```
    OutgoingTOS     : Desired Type of Service
     FlowServiceType : ControlledLoad
     MaxRate         : Target rate desired for traffic as a positive
integer
```

4. **Marking and Shaping**

```
    OutgoingTOS     : Desired Type of Service
     FlowServiceType : Guaranteed
     MaxRate         : Target rate desired for traffic as a positive
integer
```

Note:: The type of service set for the out of profile packets is set to zero in the case of policing.

Sample policyd Configuration File

The following is a complete example of the **/etc/policyd.conf** configuration file.

policyd Configuration File

```
#loglevel  511    # Verbose logging

 ######################################################################
 #
 # Mark rsh traffic on TCP source ports 513 and 514.
 ServiceCategories       tcp_513_514_svc
 {
         MaxRate                 0               # Mark only
         OutgoingTOS             00011100        # binary
         FlowServiceType         ControlledLoad
 }

 ServicePolicyRules      tcp_513_514_flt
 {
         ProtocolNumber          6  # TCP
         SourceAddressRange      0.0.0.0–0.0.0.0 # Any IP src addr
         DestinationAddressRange 0.0.0.0–0.0.0.0 # Any IP dst addr
         SourcePortRange         513–514
         DestinationPortRange    0–0             # Any dst port
         ServiceReference        tcp_513_514_svc
 }
 #
 ######################################################################
 #
 # Shape connected UDP traffic on source port 9000.
 ServiceCategories       udp_9000_svc
 {
         MaxRate                 8192     # kilobits
         MaxTokenBucket          64       # kilobits
         FlowServiceType         Guaranteed
 }

 ServicePolicyRules      udp_9000_flt
 {
         ProtocolNumber          17  # UDP
         SourceAddressRange      0.0.0.0–0.0.0.0 # Any IP src addr
         DestinationAddressRange 0.0.0.0–0.0.0.0 # Any IP dst addr
         SourcePortRange         9000–9000
         DestinationPortRange    0–0             # Any dst port
         ServiceReference        udp_9000_svc
 }
 #
 ######################################################################
 #
 # Mark and police finger traffic on TCP source port 79.
 ServiceCategories       tcp_79_svc
 {
         MaxRate                 8         # kilobits
         MaxTokenBucket          32        # kilobits
         OutgoingTOS             00011100  # binary
         FlowServiceType         ControlledLoad
 }

 ServicePolicyRules      tcp_79_flt
 {
         ProtocolNumber          6  # TCP
         SourceAddressRange      0.0.0.0–0.0.0.0 # Any IP src addr
         DestinationAddressRange 0.0.0.0–0.0.0.0 # Any IP dst addr
         SourcePortRange         79–79
         DestinationPortRange    0–0             # Any dst port
         ServiceReference        tcp_79_svc
 }
 #
 ######################################################################
 #
 # Mark and shape ftp–data traffic on TCP source port 20.
```

```
ServiceCategories       tcp_20_svc
{
        MaxRate                 81920        # kilobits
        MaxTokenBucket          128          # kilobits
        OutgoingTOS             00011101     # binary
        FlowServiceType         Guaranteed
}

ServicePolicyRules      tcp_20_flt
{
        ProtocolNumber          6  # TCP
        SourceAddressRange      0.0.0.0–0.0.0.0 # Any IP src addr
        DestinationAddressRange 0.0.0.0–0.0.0.0 # Any IP dst addr
        SourcePortRange         20–20
        DestinationPortRange    0–0             # Any dst port
        ServiceReference        tcp_20_svc
}
#
######################################################################
#
# LDAP server entry.
#ReadFromDirectory
#{
#   LDAP_Server             9.3.33.138  # IP address of LDAP server
#   Base                    o=ibm,c=us  # Base distinguished name
#   LDAP_SelectedTag        myhost      # Typically client hostname
#}
#
######################################################################
```

Loading Policies into SecureWay Directory Server

If the policy daemon is used with the SecureWay Directory LDAP Server, use the following schema as a guide to update **/etc/ldapschema/V3.modifiedschema** before starting the LDAP server. Refer to Planning and Configuration for LDAP Name Resolution (SecureWay Directory Schema) on page 4-91 for details.

LDAP Schema

```
objectClasses {
 ( ServiceCategories–OID NAME ’ServiceCategories’ SUP top MUST
 ( objectClass $ SelectorTag $ serviceName ) MAY
 ( description $ FlowServiceType $ MaxRate $ MaxTokenBucket $ OutgoingTos )
)
 ( ServicePolicyRules–OID NAME ’ServicePolicyRules’ SUP top MUST
 ( objectClass $ PolicyName $ SelectorTag ) MAY
 ( description $ DestinationAddressRange $ DestinationPortRange $
 ProtocolNumber $ ServiceReference $ SourceAddressRange $ SourcePortRange )
)
 }
 attributeTypes {
 ( DestinationAddressRange–OID NAME ’DestinationAddressRange’ SYNTAX
 1.3.6.1.4.1.1466.115.121.1.15 SINGLE–VALUE )
 ( DestinationPortRange–OID NAME ’DestinationPortRange’ SYNTAX
  1.3.6.1.4.1.1466.115.121.1.15 SINGLE–VALUE )
 ( FlowServiceType–OID NAME ’FlowServiceType’
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE–VALUE )
 ( MaxRate–OID NAME ’MaxRate’ SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
SINGLE–VALUE )
 ( MaxTokenBucket–OID NAME ’MaxTokenBucket’ SYNTAX
1.3.6.1.4.1.1466.115.121.1.15 SINGLE–VALUE )
 ( OutgoingTos–OID NAME ’OutgoingTos’ SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
SINGLE–VALUE )
 ( PolicyName–OID NAME ’PolicyName’ SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
SINGLE–VALUE )
 ( ProtocolNumber–OID NAME ’ProtocolNumber’ SYNTAX
1.3.6.1.4.1.1466.115.121.1.15 SINGLE–VALUE )
 ( SelectorTag–OID NAME ’SelectorTag’ SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
SINGLE–VALUE )
 ( ServiceReference–OID NAME ’ServiceReference’ SYNTAX
  1.3.6.1.4.1.1466.115.121.1.15 SINGLE–VALUE )
 ( SourceAddressRange–OID NAME ’SourceAddressRange’ SYNTAX
  1.3.6.1.4.1.1466.115.121.1.15 SINGLE–VALUE )
 ( SourcePortRange–OID NAME ’SourcePortRange’ SYNTAX
1.3.6.1.4.1.1466.115.121.1.15 SINGLE–VALUE )
 }

 IBMattributeTypes {
 ( DestinationAddressRange–OID DBNAME ( ’DestinationAddressRange’
’DestinationAddressRange’ ) )
 ( DestinationPortRange–OID DBNAME ( ’DestinationPortRange’
’DestinationPortRange’ ) )
 ( FlowServiceType–OID DBNAME ( ’FlowServiceType’ ’FlowServiceType’ ) )
 ( MaxRate–OID DBNAME ( ’MaxRate’ ’MaxRate’ ) )
 ( MaxTokenBucket–OID DBNAME ( ’MaxTokenBucket’ ’MaxTokenBucket’ ) )
 ( OutgoingTos–OID DBNAME ( ’OutgoingTos’ ’OutgoingTos’ ) )
 ( PolicyName–OID DBNAME ( ’PolicyName’ ’PolicyName’ ) )
 ( ProtocolNumber–OID DBNAME ( ’ProtocolNumber’ ’ProtocolNumber’ ) )
 ( SelectorTag–OID DBNAME ( ’SelectorTag’ ’SelectorTag’ ) )
 ( ServiceReference–OID DBNAME ( ’ServiceReference’ ’ServiceReference’ ) )
 ( SourceAddressRange–OID DBNAME ( ’SourceAddressRange’
’SourceAddressRange’ ) )
 ( SourcePortRange–OID DBNAME ( ’SourcePortRange’ ’SourcePortRange’ ) )
 }

 ldapSyntaxes {
 }

 matchingRules {
 }
```

System Configuration

Overlapping Policies

Policies that overlap are installed in the QoS Manager in a nondeterministic ordering. In the case of overlapping policies the `PolicyRulePriority` attribute of the `ServicePolicyRules` should be specified to determine the ordering of enforcement of policies. The `PolicyRulePriority` attribute takes an integer as a parameter and, in the case of overlapping policies, the rule with the highest integer value is enforced.

UDP usage

Only connected UDP sockets are supported for QoS.

Policy Conflicts with RSVP Reservations

The policy and RSVP agents as mutually independent. Thus, care must be taken not to specify a policy that conflicts with, or is covered by, an existing RSVP reservation. In the presence of such conflicts, the system accepts the first policy or reservation while flagging a violation for the others.

Token Bucket Depth Specification

For correct operation, the `MaxTokenBucket` attribute must be set to at least the maximum MTU of all interfaces configured in the system.

Policy Modification

Policy modifications are handled by the policy agent by automatically deleting the existing policies and installing the new ones. This may result in a short, temporary window of time during which the corresponding traffic receives default (typically best effort) service.

Standards Compliance

This release is compatible with evolving Internet Engineering Task Force (IETF) standards for Differentiated (DiffServ) and Integrated Services (IntServ) on the Internet.

IntServ Model

The following RFCs describe various components of the IntServ model:

- The Use of RSVP with IETF Integrated Services (RFC 2210)
- Specification of the Controlled–Load Network Element Service (RFC 2211)
- Specification of Guaranteed Quality of Service (RFC 2212)

DiffServ Model

The following RFCs describe various components of the DiffServ model:

- Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers (RFC 2474)
- An Architecture for Differentiated Services (RFC 2475)

The following RFC outlines the current usage of the IP TOS octet:

- Type of Service in the Internet Protocol Suite (RFC 1349)

The following RFCs outline future practices governing usage of the IP TOS octet:

- Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers (RFC 2474)
- Assured Forwarding PHB Group (RFC 2597)
- An Expedited Forwarding PHB (RFC 2598)

IPv6 Support

QoS for AIX 5.2 only supports IPv4. IPv6 is not supported.

Controlling the Policy Daemon

You can control the policy daemon by using the system resource controller (SRC). For example, the command:

```
   startsrc –s policyd –a ″–i 60″
```

starts the policy agent with a refresh interval of 60 seconds.

The command

```
   stopsrc –s policyd
```

stops the policy daemon.

Note:: Stopping the policy daemon does not remove the installed policies in the kernel. When you start the policy daemon again, the old policies (previously installed in the kernel) are deleted, and the policies defined in the **/etc/policyd.conf** file are reinstalled.

The **refresh** SRC command is not currently supported.

QoS Reference

For important updates to this documentation, consult the README file in **/usr/samples/tcpip/qos**.

Commands

- **qosadd**
- **qoslist**
- **qosmod**
- **qosremove**
- **qosstat**
- **mkqos**
- **rmqos**

Methods

- **cfgqos**
- **ucfgqos**

TCP/IP Problem Determination

This section contains information about diagnosing common problems in a Transmission Control Protocol/Internet Protocol (TCP/IP) network environment.

The **netstat** command is a good tool for determining which area of the network has a problem. Once you have isolated the problem to an area, you can use more sophisticated tools to proceed. For example, you might use the **netstat –i** and **netstat –v** to determine if you have a problem with a particular hardware interface, and then run diagnostics to further isolate the problem. Or, if the **netstat –s** command shows that there are protocol errors, you could then use the **trpt** or **iptrace** commands.

The topics discussed in this section are as follows:

- Communication Problems on page 4-231
- Name Resolution Problems on page 4-231
- Routing Problems on page 4-232
- Problems with SRC Support on page 4-234
- telnet or rlogin Problems on page 4-235
- Configuration Problems on page 4-237
- Common Problems with Network Interfaces on page 4-237
- Problems with Packet Delivery on page 4-240
- Problems with Dynamic Host Configuration Protocol (DHCP) on page 4-241

Communication Problems

If you cannot communicate with a host on your network:

- Try to contact the host, using the **ping** command. Run the **ping** command on the local host to verify that the local interface to the network is up and running.
- Try to resolve the name of the host, using the **host** command. If the name does not resolve, you have a name resolution problem. See Name Resolution Problems for more information.

If the name resolves and you are trying to contact a host on another network, you may have a routing problem. See Routing Problems for more information.

- If your network is a token–ring network, check to see if the target host is on another ring. If so, the allcast field is probably set incorrectly. Use the Web–based System Manager, **wsm**, or the System Management Interface Tool (SMIT) fast path **smit chinet** to access the Network Interfaces menu. Then, set the Confine Broadcast to Local Ring field to **no** in the token–ring dialog.
- If there are a large number of Address Resolution Protocol (ARP) packets on your network, verify that your subnet mask is set correctly. This condition is known as a broadcast storm and can affect your system performance.

Name Resolution Problems

Resolver routines on hosts running TCP/IP attempt to resolve names, using the following sources in the order listed:

1. DOMAIN name server (**named**)
2. Network Information Service (NIS)
3. Local **/etc/hosts** file

When NIS+ is installed, lookup preferences are set using the **irs.conf** file. For more information, see *AIX 5L Version 5.2* Network Information Service *(NIS and NIS+) Guide*.

Client Host

If you cannot get a host name resolved, and you are using flat name resolution (using the **/etc/hosts** file), verify that the host name and correct Internet Protocol (IP) address information is in the **/etc/hosts** file.

If you cannot get a host name resolved, and you are using a name server:

1. Verify that you have a **resolv.conf** file specifying the domain name and Internet address of a name server.
2. Verify that the local name server is up by issuing the **ping** command with the IP address of the name server (found in the local **resolv.conf** file).
3. If the local name server is up, verify that the **named** daemon on your local name server is active by issuing the **lssrc –s named** command on the name server.
4. If you are running the **syslogd**, check for logged messages. The output for these messages is defined in the **/etc/syslog.conf** file.

If these steps do not identify the problem, check the name server host.

Name Server Host

If you cannot get a host name resolved:

1. Verify that the **named** daemon is active by issuing the following command:

   ```
   lssrc –s named
   ```

2. Verify that the address of the target host exists and is correct in the name server database. Send a **SIGINT** signal to the **named** daemon to dump the database and cache to the file **/var/tmp/named_dump.db**. Verify that the address you are trying to resolve is there and is correct.

   Add or correct name–to–address resolution information in the **named** hosts data file for the master name server of the domain. Then issue the following **SRC** command to reread the data files:

   ```
   refresh –s named
   ```

3. Verify that the name resolution requests are being processed. To do this, enter the **named** daemon from the command line and specify a debugging level. Valid debug levels are 1 through 9. The higher the level, the more information the debug mechanism logs.

   ```
   startsrc –s named –a ”–d   DebugLevel  ”
   ```

4. Check for configuration problems in the **named** data files. For more information, see Name Server Overview on page 4-74. In addition, see the ”DOMAIN Data File Format,”, ”DOMAIN Reverse Data File Format,” ”DOMAIN Cache File Format,” and the ”DOMAIN Local Data File Format” in the *AIX 5L Version 5.2 Files Reference*.

   Note:: A common error is the incorrect use of the . (period) and the @ (at sign) in the DOMAIN data files.

If external users cannot reach your domains:

- Make sure that all your non–master name servers (slave, hint) have equal time–to–live (TTL) information in the DOMAIN data files.

If external resolvers query your servers constantly:

- Make sure your servers are distributing DOMAIN data files with reasonable TTL values. If the TTL is zero or another small value, the data you transfer times out very quickly. Set the minimum value in your start of authority (SOA) records to a week or more to solve this problem.

Routing Problems

If you cannot reach a destination host, consider the following situations:

- If you receive a `Network Unreachable` error message, make sure that a route to the gateway host has been defined and is correct. Check this by using the **netstat –r** command to list kernel routing tables.
- If you receive a `No route to host` error message, verify that the local network interface is up by issuing the **ifconfig** interface_name command. The output indicates whether or not the interface is up. Use the **ping** command to try and reach another host on your network.
- If you receive a `Connection timed out` error message:
  - Verify that the local gateway is up using the **ping** command with the name or Internet address of the gateway.
  - Make sure that a route to the gateway host has been defined and is correct. Check this by using the **netstat –r** command to list kernel routing tables.
  - Make sure the host you want to communicate with has a routing table entry back to your machine.
- If you are using static routing, make sure that a route to the target host and gateway host has been defined. Check this by using the **netstat –r** command to list kernel routing tables.

  Note:: Make sure the host you want to communicate with has a routing table entry to your machine.
- If you are using dynamic routing, verify that the gateway is listed and correct in the kernel routing tables by issuing the **netstat –r** command.
- If the gateway host is using the **Routing Information Protocol** (RIP) with the **routed** daemon, make sure that a static route to the target host is set up in the **/etc/gateways** file.

  Note:: You need to do this only if the routing daemon cannot identify the route to a distant host through queries to other gateways.
- If the gateway host is using the **RIP** with the **gated** daemon, make sure that a static route to the target host is set up in the **gated.conf** file.
- If you are using dynamic routing with the **routed** daemon:
  - If **routed** cannot identify the route through queries (for example, if the target host is not running the **RIP**, check the **/etc/gateways** file to verify that a route to the target host is defined.
  - Make sure that gateways responsible for forwarding packets to the host are up and running the **RIP**. Otherwise, you'll need to define a static route.
  - Run the **routed** daemon using the debug option to log such information as bad packets received. Invoke the daemon from the command line using the following command:

    ```
    startsrc –s routed –a ”–d”
    ```
  - Run the **routed** daemon using the **–t** flag, which causes all packets sent or received to be written to standard output. When **routed** is run in this mode, it remains under the control of the terminal that started it. Therefore, an interrupt from the controlling terminal kills the daemon.
- If you are using dynamic routing with the **gated** daemon:
  - Verify that the **/etc/gated.conf** file is configured correctly and that you are running the correct protocols.
  - Make sure the gateway on the source network is using the same protocol as the gateway on the destination network.

  - Make sure that the machine with which you are trying to communicate has a route back to your host machine.
  - Verify that the gateway names in the **gated.conf** file correspond to the gateway names listed in the **/etc/networks** file.
- If you are using the **RIP** or **HELLO** protocols, and routes to the destination cannot be identified through routing queries, check the **gated.conf** file to verify that a route to the target host is defined. Set static routes under the following conditions:
  - The destination host is not running the same protocol as the source host so cannot exchange routing information.
  - The host must be reached by a distant gateway (a gateway that is on a different autonomous system than the source host). The **RIP** can be used only among hosts on the same autonomous system.

Other Possibilities

If all else fails, you might want to turn on tracing for your routing daemon (either **routed** or **gated**). Use the SRC **traceson** command from the command line, or send a signal to the daemon to specify different levels of tracing. See the **gated** daemon or the **routed** daemon for specifics on sending signals to these daemons.

Problems with SRC Support

- If changes to the **/etc/inetd.conf** file do not take effect:

  Update the **inetd** daemon by issuing the **refresh –s inetd** command or the **kill –1** *InetdPID* command.

- If the **startsrc –s** [*subsystem name*] returns the following error message:

  ```
  0513–00  The System Resource Controller is not active.
  ```

  The System Resource Controller subsystem has not been activated. Issue the **srcmstr &** command to start SRC, then reissue the **startsrc** command.

  You might also want to try starting the daemon from the command line without SRC support.

- If the **refresh –s** [*subsystem name*] or **lssrc –ls** [*subsystem name*] returns the following error message:

  ```
  [subsystem name] does not support this option.
  ```

  The subsystem does not support the SRC option issued. Check the subsystem documentation to verify options the subsystem supports.

- If the following message is displayed:

  ```
  SRC was not found, continuing without SRC support.
  ```

  A daemon was invoked directly from the command line instead of using the **startsrc** command. This is not a problem. However, SRC commands, such as **stopsrc** and **refresh,** will not manipulate a subsystem that is invoked directly.

If the **inetd** daemon is up and running correctly and the appropriate service seems to be correct but you still cannot connect, try running the **inetd** daemon processes through a debugger.

1. Stop the **inetd** daemon temporarily:

   ```
   stopsrc –s inetd
   ```

   The **stopsrc** command stops subsystems like the **inetd** daemon.

2. Edit the **syslog.conf** file to add a debugging line at the bottom. For example:

   ```
   vi /etc/syslog.conf
   ```

   a. Add the line " `*.debug /tmp/myfile` " at the bottom of the file and exit.

   b. The file that you specify must exist (`/tmp/myfile` in this example). You can use the **touch** command to make your file exists.

3. Refresh the file:

   – If you are using SRC, enter:

   ```
   refresh –s syslogd
   ```

   – If you are not using SRC, kill the **syslogd** daemon:

   ```
   kill –1 ‘ps –e | grep /etc/syslogd | cut –c1–7‘
   ```

4. Start the **inetd** daemon backup with debugging enabled:

   ```
   startsrc –s inetd –a ″–d″
   ```

   The **–d** flag enables debugging.

5. Try to make a connection to log errors in the `/tmp/myfile` debugging file. For example:

   ```
   tn bastet
    Trying...
    connected to bastet
    login:>
    Connection closed
   ```

6. See if anything shows up as a problem in the debugging file. For example:

   ```
   tail –f /tmp/myfile
   ```

telnet or rlogin Problems

The following explanations can be useful in solving problems with the **telnet** or **rlogin** command.

Screen Distortion

If you are having trouble with screen distortion in full–screen applications:

1. Check the **TERM** environment variable by issuing one of the following commands:

   ```
   env
   ```

```
echo $TERM
```

2. Verify that the **TERM** variable is set to a value that matches the type of terminal display you are using.

telnet Debugging

telnet subcommands that can help in debugging problems include:

display	Displays set and toggle values.
toggle	Toggles the display of all network data in hex.
toggle options	Toggles the display of internal **telnet** process options.

telnetd Daemon Debugging

If the **inetd** daemon could execute the **telnet** service but you still cannot connect using the **telnet** command, there may be something wrong with the **telnet** interface.

1. Verify that **telnet** is using the correct terminal type.

   a. Check the **$TERM** variable on your machine:

   ```
   echo $TERM
   ```

   b. Log in to the machine to which you are trying to attach and check the **$TERM** variable:

   ```
   echo $TERM
   ```

2. Use the **telnet** interface's debugging capabilities by entering the **telnet** command without flags.

   ```
   telnet
    tn>
   ```

   a. Enter `open host` where *host* is the name of the machine.

   b. Enter Ctrl–T to get to the `tn%gt;` prompt.

   c. At the `tn>` prompt, enter `debug` for debugging mode.

3. Try to connect to another machine using the **telnet** interface:

   ```
   telnet bastet
    Trying...
    Connected to bastet
    Escape character is '^T'.
   ```

   Watch the display as the various commands scroll up the screen. For example:

   ```
   SENT do ECHO
   SENT do SUPPRESS GO AHEAD
   SENT will TERMINAL TYPE (reply)
   SENT do SUPPORT SAK
   SENT will SUPPORT SAK (reply)
   RCVD do TERMINAL TYPE (don't reply)
   RCVD will ECHO (don't reply)
   RCVD will SUPPRESS GO AHEAD (don't reply)
   RCVD wont SUPPORT SAK (reply)
   SENT dont SUPPORT SAK (reply)
   RCVD do SUPPORT SAK (don't reply)
   SENT suboption TELOPT_NAWS Width 80, Height 25
   RCVD suboption TELOPT_TTYPE SEND
   RCVD suboption TELOPT_TTYPE aixterm
   ...
   ```

4. Check **/etc/termcap** or **/usr/lib/terminfo** for the `aixterm` definition. For example:

   ```
   ls –a /usr/lib/terminfo
   ```

5. If the `aixterm` definition is missing, add it by building the **ibm.ti** file. For example:

   ```
   tic ibm.ti
   ```

   The **tic** command is a terminal information compiler.

Programs Using Extended Curses

Problems with function and arrow keys can arise when using the **rlogin** and **telnet** commands with programs using extended curses. Function and arrow keys generate escape sequences, which are split if too little time is allotted for the entire key sequence. Curses waits a specific amount of time to decide whether an Esc indicates the escape key only or the start of a multibyte escape sequence generated by other keys, such as cursor keys, the action key, and function keys.

If no data, or data that is not valid, follows the Esc in the allotted amount of time, curses decides that the Esc is the escape key, and the key sequence is split. The delay resulting from the **rlogin** or **telnet** command is network dependent. Sometimes arrow and function keys work and sometimes they do not, depending on the speed of the network to which you are connecting. Setting the **ESCDELAY** environment variable to a large value (1000 to 1500) effectively solves this problem.

Configuration Problems

Network interfaces are automatically configured during the first system startup after the adapter card is installed. However, you still need to set some initial values for TCP/IP including the host name, the Internet address, and the subnet mask. To do this, you can use the Web–based System Manager, **wsm**, or you can use the SMIT interface in the following ways:

- Use the **smit mktcpip** fast path to set the initial values for the host name, the Internet address, and the subnet mask.
- Use the **smit mktcpip** fast path to specify a name server to provide name resolution service. (Note that **smit mktcpip** configures one network interface only.)
- Use the **smit chinet** fast path to set other network attributes.

You may also want to set up any static routes the host needs for sending transmitting information, such as a route to the local gateway. Use the Web–based System Manager, **wsm**, or the SMIT fast path, **smit mkroute**, to set these up permanently in the configuration database.

If you are having other problems with your configuration, see the Configuring a TCP/IP Network Checklist for more information.

Common Problems with Network Interfaces

Network interfaces are configured automatically during the first system startup after the adapter card is installed. However, there are certain values that must be set in order for TCP/IP to start. These include the host name and Internet address and can be set using the Web–based System Manager, **wsm**, or the SMIT fast path, **smit mktcpip**.

If you choose the SMIT method, use the **smit mktcpip** fast path to set these values permanently in the configuration database. Use the **smit chinet** and **smit hostname** fast paths to change them in a running system. The **smit mktcpip** fast path minimally configures TCP/IP. To add adapters, use the Further Configuration menu, which can be reached with the **smit tcpip** fast path.

If you have already checked these to verify accuracy and you are still having trouble sending and receiving information, check the following:

- Verify that your network adapter has a network interface by executing the **netstat –i** command. The output should list an interface, such as **tr0**, in the `Name` column. If it does not, create a network interface through Web-based System Manager or by entering the SMIT fast path **smit mkinet**.
- Verify that IP address for the interface is correct by executing the **netstat –i** command. The output should list the IP address in the Network column. If it is incorrect, set the IP address through Web-based System Manager or by entering the SMIT fast path **smit chinet**.
- Use the **arp** command to make sure you have the complete IP address for the target machine. For example:

  ```
  arp –a
  ```

  The **arp** command looks for the physical adapter address. This command might show an incomplete address. For example:

  ```
  ? (192.100.61.210) at (incomplete)
  ```

  This could be due to an unplugged machine, a stray address with no machine at that particular address, or a hardware problem (such as a machine that connects and receives packets but is not able to send packets back).

- Look for errors on the adapter card. For example:

  ```
  netstat –v
  ```

  The **netstat –v** command shows statistics for the Ethernet, Token Ring, X.25, and 802.3 adapter device drivers. The command also shows network and error logging data for all device drivers active on an interface including: `No Mbufs Errors`, `No Mbuf Extension Errors`, and `Packets Transmitted and Adapter Errors Detected`.

- Check the error log by running the **errpt** command to ensure that there are no adapter problems.
- Verify that the adapter card is good by running diagnostics. Use the Web-based System Manager Devices application, the **smit diag** fast path, or the **diag** command.

If these steps do not identify the problem, see Problems with a SLIP Network Interface on page 4-238,

Problems with an Ethernet Network Interface on page 4-238, or Problems with a Token–Ring Network Interface.

Problems with a SLIP Network Interface

In general, the most effective method for debugging problems with a Serial Line Interface Protocol (SLIP) interface is to retrace your configuration, verifying each step. However, you can also:

- Verify that the **slattach** process is running and using the correct tty port by issuing the **ps –ef** command. If it is not, run the **slattach** command. (See Configuring SLIP over a Modem or Configuring SLIP over a Null Modem Cable for the exact syntax you should use.)
- Verify that the point–to–point addresses are specified correctly by entering the **smit chinet** fast path.

  Select the SLIP interface. Make sure that the INTERNET ADDRESS and DESTINATION Address fields are correct.

If the modem is not functioning correctly:

- Make sure that the modem was installed correctly. See the modem installation manual.
- Verify that any flow control the modem does is turned off.

If the tty is not functioning correctly, verify that the tty baud rate and modem characteristics are set correctly in the configuration database by entering the **smit tty** fast path.

Problems with an Ethernet Network Interface

If the network interface has been initialized, the addresses correctly specified, and you have verified that the adapter card is good:

- Verify that you are using a T–connector plugged directly into the inboard/outboard transceiver.
- Make sure you are using an Ethernet cable. (Ethernet cable is 50 OHM.)
- Make sure you are using Ethernet terminators. (Ethernet terminators are 50 OHM.)
- Ethernet adapters can be used with either the transceiver that is on the card or with an external transceiver. There is a jumper on the adapter to specify which you are using. Verify that your jumper is set correctly (see your adapter manual for instructions).
- Verify that you are using the correct Ethernet connector type (thin is BNC; thick is DIX). If you change this connector type, use the Web–based System Manager, **wsm**, or the SMIT fast path, **smit chgenet**, to set the Apply Change to Database Only field. (Check the field in Web-based System Manager or set to **yes** in SMIT.) Restart the machine to

apply the configuration change. (See Configuring and Managing Adapters on page 4-36.)

Problems with a Token–Ring Network Interface

If you cannot communicate with some of the machines on your network although the network interface has been initialized, the addresses correctly specified, and you have verified that the adapter card is good:

- Check to see if the hosts with whom you cannot communicate are on a different ring. If they are, use the Web–based System Manager, **wsm**, or the SMIT fast path **smit chinet** to check the Confine BROADCAST to Local Token–Ring field. *Do not* check the field in Web-based System Manager or set to **no** in SMIT.
- Check to see whether the token–ring adapter is configured to run at the correct ring speed. If it is configured incorrectly, use the Web-based System Manager Network application or SMIT to change the adapter ring speed attribute (see Configuring and Managing Adapters). When TCP/IP is restarted, the token–ring adapter has the same ring speed as the rest of the network.

Problems with a Token–Ring/Ethernet Bridge

If you cannot communicate between a token–ring and an Ethernet network, using a bridge, and you have verified that the bridge is functioning correctly, the Ethernet adapter might be dropping packets. A machine drops packets if the incoming packet (including headers) is greater than the network adapter maximum transmission unit (MTU) value. For instance, a 1500–byte packet sent by a token–ring adapter over the bridge collects an 8–byte logical link control (LLC) header, making the total packet size 1508. If the receiving Ethernet adapter MTU is set to 1500, the packet is dropped.

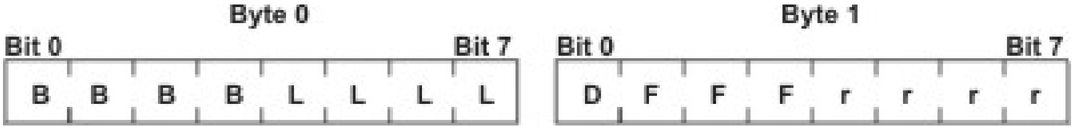
Check the MTU values of both network adapters. To allow for the eight–byte LLL header, the token–ring adapter attaches to outgoing packets, set the MTU value for the token–ring adapter at least eight bytes lower than the MTU value for the Ethernet adapter. For example, set the MTU for a token–ring adapter to 1492 to communicate with an Ethernet adapter with an MTU of 1500.

Problems with a Token–Ring/Token–Ring Bridge

When operating through a bridge, change the default value of 1500 for the maximum transmission unit (MTU) to a value that is eight less than the maximum information field (maximum I–frame) advertised by the bridge in the routing control field.

To find the routing control field value, use the **iptrace** daemon to look at incoming packets. Bits 1, 2, and 3 of Byte 1 are the Largest Frame Bits, which specify the maximum information field that can be transmitted between two communicating stations on a specific route. See the following for the format of the routing control field:

Figure 26. Routing Control Field This illustration shows byte 0 and byte 1 of a routing control field. The eight bits of byte one are B, B, B, B, L, L, L, L. The eight bits of byte 1 are D, F, F, F, r, r, r, r.



Values for the Largest Frame Bits are as follows:

000 Specifies a maximum of 516 bytes in the information field.

001 Specifies a maximum of 1500 bytes in the information field.

010 Specifies a maximum of 2052 bytes in the information field.

011	Specifies a maximum of 4472 bytes in the information field.
100	Specifies a maximum of 8144 bytes in the information field.
101	Reserved.
110	Reserved.
111	Used in all–routes broadcast frames.

For example, if the maximum I–frame value is 2052 in the routing control field, the MTU size should be set to 2044. This is for token–ring network interfaces only.

Note:: When using **iptrace**, the output file must *not* be on a Network File System (NFS).

Problems with Packet Delivery

Communicating with a Remote Host

If you cannot communicate with a remote host, try the following:

- Run the **ping** command on the local host to verify that the local interface to the network is up and running.
- Use the **ping** command for hosts and gateways that are progressively more hops from the local host to determine the point at which communication fails.

If you are having trouble with packet loss or are experiencing delays in packet delivery, try the following:

- Use the **trpt** command to trace packets at the socket level.
- Use the **iptrace** command to trace all protocol layers.

If you cannot communicate between a token–ring and an Ethernet network using a bridge, and you have verified that the bridge is good:

- Check the MTU values of both adapters. The MTU values must be compatible to allow communication. A machine drops packets if the incoming packet (including headers) is greater than the adapter's MTU values. For instance, a 1500–byte packet sent over the bridge collects an 8–byte LLC header, making the total packet size 1508. If the receiving machine MTU is set to 1500, a packet of 1508 bytes is dropped.

snmpd Response to Queries

If **snmpd** is not responding to queries and there are no log messages received, the packet might be to large for the kernel User Datagram Protocol (UDP) packet handler. If this is the case, increase the kernel variables, **udp_sendspace** and **udp_recvspace** by issuing the following commands:

```
no –o udp_sendspace=64000
 no –o udp_recvspace=64000
```

The maximum size for a UPD packet is 64K. If your query is larger than 64K, it will be rejected. Split the packet into smaller packets to avoid this problem.

Problems with Dynamic Host Configuration Protocol (DHCP)

If you cannot get an IP address or other configuration parameters:

- Check to see that you have specified an interface to be configured. This can be done through the Web-based System Manager Network application, by editing the **/etc/dhcpcd.ini** file, or by using the SMIT fast path **smit dhcp**.
- Check to see that there is a server on the local network or a relay agent configured to get your requests off the local network.
- Check to see that the **dhcpcd** program is running. If it is not, use the **startsrc –s dhcpcd** command.

TCP/IP Reference

Transmission Control Protocol/Internet Protocol (TCP/IP) topics discussed in this section are:

- List of TCP/IP Commands on page 4-242
- List of TCP/IP Daemons on page 4-243
- List of Methods on page 4-243
- List of TCP/IP Files on page 4-243
- List of RFCs on page 4-244

List of TCP/IP Commands

chnamsv	Changes Transmission Control Protocol/Internet Protocol (TCP/IP) based name service configuration on a host.
chprtsv	Changes a print service configuration on a client or server machine.
hostent	Directly manipulates address–mapping entries in the system configuration database.
ifconfig	Configures or displays network interface parameters for a network, using TCP/IP.
mknamsv	Configures TCP/IP–based name service on a host for a client.
mkprtsv	Configures TCP/IP–based print service on a host.
mktcpip	Sets the required values for starting TCP/IP on a host.
no	Configures network options.
rmnamsv	Unconfigures TCP/IP–based name service on a host.
rmprtsv	Unconfigures a print service on a client or server machine.
slattach	Attaches serial lines as network interfaces.
arp	Displays or changes the Internet address to hardware address translation tables used by the Address Resolution Protocol (ARP).
gettable	Gets Network Information Center (NIC) format host tables from a host.
hostid	Sets or displays the identifier of the current local host.
hostname	Sets or displays the name of the current host system.
htable	Converts host files to the format used by network library routines.
ipreport	Generates a packet trace report from the specified packet trace file.
iptrace	Provides interface–level packet tracing for Internet protocols.
lsnamsv	Shows name service information stored in the database.
lsprtsv	Shows print service information stored in the database.
mkhosts	Generates the host table file.
namerslv	Directly manipulates domain name server entries for local resolver routines in the system configuration database.
netstat	Shows network status.
route	Manually manipulates the routing tables.
ruser	Directly manipulates entries in three separate system databases that control foreign host access to programs.

ruptime	Displays the status of each host on a network.
securetcpip	Enables the network security feature.
setclock	Sets the time and date for a host on a network.
timedc	Returns information about the **timed** daemon.
trpt	Performs protocol tracing on Transmission Control Protocol (TCP) sockets.

List of TCP/IP Daemons

fingerd	Provides remote user information.
ftpd	Provides the server function for the Internet File Transfer Protocol (FTP) protocol.
gated	Provides gateway routing functions for the Routing Information Protocol (RIP), Hello Protocol (HELLO), Exterior Gateway Protocol (EGP), Border Gateway Protocol (BGP), and Simple Network Management Protocol SNMP).
inetd	Provides Internet service management for a network.
named	Provides the server function for the Domain Name Protocol (DOMAIN).
rexecd	Provides the server function for the **rexec** command.
rlogind	Provides the server function for the **rlogin** command.
routed	Manages network routing tables.
rshd	Provides the server function for remote command execution.
rwhod	Provides the server function for the **rwho** and **ruptime** commands.
syslogd	Reads and logs system messages.
talkd	Provides the server function for the **talk** command.
telnetd	Provides the server function for the TELNET protocol.
tftpd	Provides the server function for the Trivial File Transfer Protocol (TFTP).
timed	Invokes the **timeserver** daemon at system startup time.

List of Methods

Device methods are programs associated with a device that perform basic device configuration operations. See List of TCP/IP Programming References in in *AIX 5L Version 5.2 Communications Programming Concepts* for information about TCP/IP methods.

List of TCP/IP Files

/etc/rc.bsdnet

See List of TCP/IP Programming References in in *AIX 5L Version 5.2 Communications Programming Concepts* for information about TCP/IP files and file formats.

List of RFCs

For a list of the RFCs (Request for Comments) supported by this operating system, see the List of TCP/IP Programming References in in *AIX 5L Version 5.2 Communications Programming Concepts*.

- RFC 1359 *Connecting to the Internet: What connecting institutions should anticipate*
- RFC 1325 *FYI on questions and answers: Answers to commonly asked ’new Internet user’ questions*
- RFC 1244 *Site Security Handbook*
- RFC 1178 *Choosing a Name for Your Computer*
- RFC 1173 *Responsibilities of host and network managers: A summary of the ‘oral tradition’ of the Internet*

Chapter 5. Network Management

The Network Management facility provides comprehensive management of system networks through the use of Simple Network Management Protocol (SNMP) enabling network hosts to exchange management information. SNMP is an internetworking protocol designed for use with TCP/IP–based internets. The following sections provide information to assist the network manager in understanding and working with SNMP:

- SNMP for Network Management on page 5-2
- SNMPv3 on page 5-3
- SNMPv1 on page 5-13

When AIX 5.2 is installed, the SNMPv3 non–encryptd version will be installed by default and will be started at system boot time. If you had your own communities, traps, and smux entries configured in your **/etc/snmpd.conf** file, you will need to migrate those manually to the **/etc/snmpdv3.conf** file. For information on migrating the communities, see Migrate from SNMPv1 to SNMPv3 on page 1-7.

You may also want to consult the information in SNMP Overview for Programmers in *AIX 5L Version 5.2 Communications Programming Concepts*.

SNMP for Network Management

SNMP network management is based on the familiar client/server model that is widely used in TCP/IP–based network applications. Each host that is to be managed runs a process called an *agent*. The agent is a server process that maintains the Management Information Base (MIB) database for the host. Hosts that are involved in network management decision–making can run a process called a manager. A *manager* is a client application that generates requests for MIB information and processes responses. In addition, a manager may send requests to agent servers to modify MIB information.

SNMP in AIX provides support for the following RFCs:

RFC 1155	Structure and Identification of Management Information for TCP/IP–based Internets
RFC 1157	A Simple Network Management Protocol (SNMP)
RFC 1213	Management Information Base for Network Management of TCP/IP–based internets: MIB–II
RFC 1227	Simple Network Management Protocol (SNMP) single multiplexer (SMUX) protocol and Management Information Base (MIB)
RFC 1229	Extensions to the generic interface Management Information Base (MIB)
RFC 1231	IEEE 802.5 token–ring Management Information Base (MIB)
RFC 1398	Definitions of Managed Objects for the Ethernet–like Interface Types
RFC 1512	FDDI Management Information Base
RFC 1514	Host Resources MIB
RFC 1592	Simple Network Management Protocol–Distributed Program Interface Version 2
RFC 1905	Protocol Operations for Version 2 of the Simple Network Management Protocol (SNMPv2)
RFC 1907	Management Information Base for Version 2 of the Simple Network Management Protocol (SNMPv2)
RFC 2572	Message Processing and Dispatching for the Simple Network Management Protocol (SNMP)
RFC 2573	SNMP Applications
RFC 2574	User–based Security Model (USM) for version 3 of the Simple Network Management Protocol (SNMPv3)
RFC 2575	View–based Access Control Model (VACM) for the Simple Network Management Protocol (SNMP)

SNMPv3

The information in this section applies to SNMPv3 only. The following secions are included in this section:

- SNMPv3 Introduction on page 5-4
- SNMPv3 Architecture on page 5-5
- SNMPv3 User Keys on page 5-7
- Issuing SNMPv3 requests on page 5-10
- Troubleshooting SNMPv3 on page 5-11

In addition, the following SNMPv3 tasks are included:

- Migrate from SNMPv1 to SNMPv3 on page 1-7
- Create Users in SNMPv3 on page 1-11
- Dynamically update authentication and privacy keys in SNMPv3 on page 1-16

SNMPv3 Introduction

Previous to AIX 5.2, SNMPv1 was the only available version of SNMP for AIX. SNMPv3 is provoded in AIX 5.2. SNMPv3 provides a powerful and flexible framework for message security and access control. Message security involves providing the following:

- Data integrity checking to ensure that the data was not altered in transit
- Data origin verification to ensure that the request or response originates from the source that it claims to have come from
- Message timeliness checking and, optionally, data confidentiality to protect against eavesdropping

The SNMPv3 architecture introduces the User–based Security Model (USM) for message security and the View–based Access Control Model (VACM) for access control. The architecture supports the concurrent use of different security, access control, and message processing models. For example, community–based security can be used concurrently with USM, if desired.

USM uses the concept of a user for which security parameters (levels of security, authentication and privacy protocols, and keys) are configured at both the agent and the manager. Messages sent using USM are better protected than messages sent with community–based security, where passwords are sent in the clear and diplayed in traces. With USM, messages exchanged between the manager and the agent have data integrity checking and data origin authentication. Message delays and message replays (beyond what happens normally due to a connection–less transport protocol) are protected against with the use of time indicators and request IDs. Data confidentiality, or encryption, is also available, where permitted, as a separately installable product. The SNMP encrypted version can be found on the AIX Expansion Pack.

The use of VACM involves defining collections of data (called views), groups of users of the data, and access statements that define which views a particular group of users can use for reading, writing, or receipt in a trap.

SNMPv3 also introduces the ability to dynamically configure the SNMP agent using SNMP SET commands against the MIB objects that represent the agent's configuration. This dynamic configuration support enables addition, deletion, and modification of configuration entries either locally or remotely.

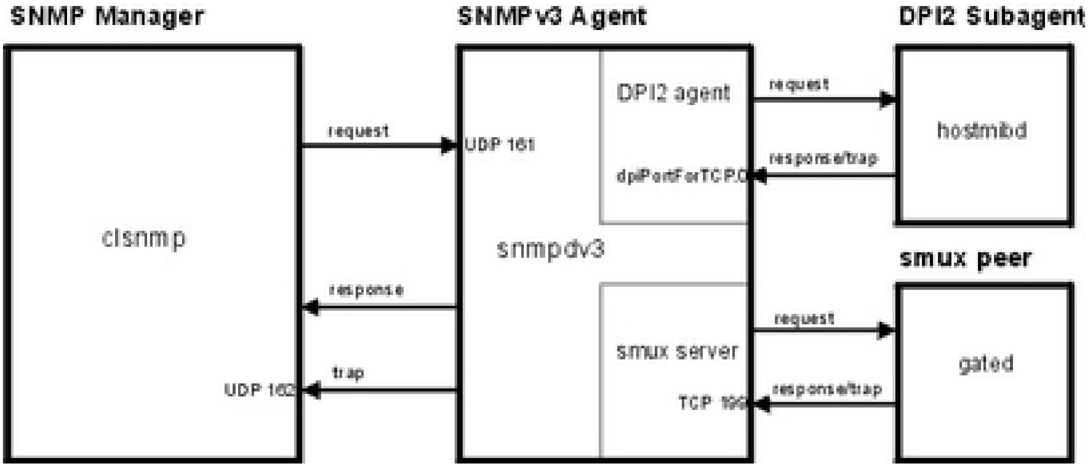
SNMPv3 access policies and security parameters are specified in the **/etc/snmpdv3.conf** file on the SNMP agent and **/etc/clsnmp.conf** file on the SNMP manager. For a scenario on how to configure these files, see Create Users in SNMPv3 on page 1-11. You can also refer to the **/etc/snmpdv3.conf** and **/etc/clsnmp.conf** file formats in *AIX 5L Version 5.2 Files Reference*.

SNMPv3 Architecture

There are four main parts to the SNMPv3 architecture as shown in the following graphic. How these systems interact with each other to provide the necessary data requested is described in this section.

Figure 27. The primary parts of the SNMPv3 architecture

This illustration shows and example of the SNMPv3 architecture. The DPI2 subagent, smux peer, SNMP manager, and SNMP agent are shown. In addition, how they communicate with each other is shown.



SNMP Agent

The SNMP agent receives requests from and makes responses to the SNMP manager. In addition, the SNMP agent communicates with all DPI2 subagents and smux peers on the system. The SNMP agent manages some MIB variables, and all DPI2 subagents and smux peers register their MIB variables with the SNMP agent.

When **clsnmp** (the SNMP manager) issues a request, it is sent to UDP 161 on the SNMP agent. If the request is an SNMPv1 or SNMPv2c request, the SNMP agent will verify the community name and process the request. If the request is an SNMPv3 request, the SNMP agent will attempt to authenticate the user requesting the data and ensure that the user has the access permissions required to fulfill the request by using the authentication keys, and, if the encrypted version is running, privacy keys. If the SNMP agent cannot authenticate the user, or if the user does not have the correct access permissions to fulfill the request, the SNMP agent will not honor the request. For information on creating users in SNMPv3, see Create Users in SNMPv3.

If the user is authenticated and has the correct access permissions, the SNMP agent will fulfill the request. The SNMP agent will locate the MIB variables being requested. If the SNMP agent itself is managing the requested MIB variables, it will process the request and send a response back to the SNMP manager. If a DPI2 subagent or smux peer is managing the requested MIB variables, the SNMP agent will forward the request to the DPI2 subagent or smux peer on which the MIB variables are managed, allow it to process the request, and will then respond to the SNMP manager.

DPI2 Subagents

A DPI2 subagent, such as **hostmibd**, communicates with the DPI2 agent, which, in SNMPv3, is part of the SNMP agent. The DPI2 subagent sends responses and traps to the DPI2 agent through the dpiPortForTCP.0. Becasue this is not a well–know port, the DPI2 subagent must first issue a request for the port number for dpiPortForTCP.0. This request is issued to UDP 161 on the SNMP agent, after which the SNMP agent responds to the DPI2 subagent with the port number for dpiPortForTCP.0. After the port number is received, the DPI2 subagent establishes a connection with the DPI2 agent using the port number given. The DPI2 subagent then registers its MIB subtrees with the DPI2 agent.

After the connection is established and the MIB subtrees have been registered, the DPI2 subagent is ready to respond to requests received from the DPI2 agent. When a request is received, the DPI2 subagent processes the request and responds with the necessary information.

The DPI2 subagent is also ready to send traps, if necessary. When a trap is sent, the SNMP agent will check its **/etc/snmpdv3.conf** file to determine the IP address or addresses to which the trap must be forwarded to, and it will send the trap to those addresses.

Smux Peers

A smux peer, such as **gated**, when started, will establish the connection to TCP 199 and will initialize the smux association. Following the initilization, the smux peer will register the MIB subtrees it is going to manage.

After the registration, the smux peer is ready to accept any incoming request from the smux server and send responses back. When the smux peer receives a request, it will process the request and send a response back the the smux server.

The smux peer can also send a trap to the smux server. If a trap is sent, the SNMP agent will check the **/etc/snmpdv3.conf** file to determine the IP address or addresses to which the trap must be forwarded, and it will send the trap to those addresses.

SNMP Manager

The SNMP manager runs **clsnmp**, which is compatible with SNMPv1, SNMPv2c, and SNMPv3. Using the **clsnmp** command, a user can issue a request, such as a `get`, `get–next`, `get–bulk`, or `set` request. The request is sent to UDP 161 on the SNMP agent, after which it waits for the response from the SNMP agent.

It also can listen to SNMP traps on UDP 162. The SNMP manager will receive traps if its IP address is so specified in the **/etc/snmpdv3.conf** file on the SNMP agent.

MIB Variables

For information on MIB variables, see Management Information Base, Terminology Related to Management Information Base Variables, Working with Management Information Base Variables, and Management Information Base Database in *AIX 5L Version 5.2 Communications Programming Concepts*.

If you want to configure your own DPI2 subagent or smux peer, see the **/usr/samples/snmpd/smux** and **/usr/samples/snmpd/dpi2** directories.

SNMPv3 User Keys

Authentication Keys

Authentication is generally required for SNMPv3 requests to be processed (unless the security level requested is `noAuth`). When authenticating a request, the SNMP agent verifies that the authentication key sent in an SNMPv3 request can be used to create a message digest that matches the message digest created from the authentication key defined by the user.

When a request is issued from the SNMP manager, the **clsnmp** command uses the authentication key found on an entry in the **/etc/clsnmp.conf** file on the SNMP manager. It needs to correlate with the authentication key specified on a `USM_USER` entry for that user in the SNMP agent's **/etc/snmpdv3.conf** file. Authentication keys are generated using the **pwtokey** command.

The authentication key is generated from two pieces of information:

- The specified password
- The identification of the SNMP agent at which the key will be used. If the agent is an IBM agent, and its engineID was generated using the vendor–specific engineID formula, the agent may be identified by IP address or hostname. Otherwise, the engineID must be provided as the agent identification.

A key that incorporates the identification of the agent at which it will be used is called a localized key. It can be used only at that agent. A key that does not incorporate the engineID of the agent at which it will be used is called non–localized.

Keys stored in the **clsnmp** command's configuration file, **/etc/clsnmp.conf**, are expected to be non–localized keys. Keys stored in the SNMP agent's configuration file, **/etc/snmpdv3.conf**, can be either localized or non–localized, though using localized keys is considered more secure.

As an alternative to storing authentication keys in the client configuration file, the **clsnmp** command allows user passwords to be stored. If the **clsnmp** command is configured with a password, the code generates an authentication key (and a privacy key if requested, and if the encrypted version is installed) for the user. These keys must produce the same authentication values as the keys configured for the `USM_USER` in the agent's **/etc/snmpdv3.conf** file or configured dynamically with the SNMP SET commands. However, the use of passwords in the client configuration file is considered less secure that the use of keys in the configuration file.

Privacy Keys

Encryption is available as a separate product on the AIX Expansion Pack where export laws allow. Keys used for encryption are generated using the same algorithms as those used for authentication. However, key lengths may differ. For example, an HMAC–SHA authentication key is 20 bytes long, but a localized encryption key used with HMAC–SHA is only 16 bytes long.

The encrypted version is automatically activated after installation. To switch back to the non–encrypted version, use the **snmpv3_ssw** command.

Generating Keys

AIX uses the **pwtokey** command to generate authentication and, when applicable, privacy keys. The **pwtokey** command enables the conversion of passwords into localized and non–localized authentication and privacy keys. The **pwtokey** procedure takes a password and an identifier as the agent and generates authentication and privacy keys. Because the procedure used by the **pwtokey** command is the same algorithm used by the **clsnmp** command, the person configuring the SNMP agent can generate appropriate authentication (and privacy) keys to put into the **/etc/clsnmp.conf** file on the SNMP manager for a user, given a particular password and the IP address at which the target will run on.

Once you have generated the authentication keys (and privacy keys if you are running the encrypted version), you will need to enter those keys in the the **/etc/snmpdv3.conf** file on the SNMP agent and in the **/etc/clsnmp.conf** file on the SNMP manager.

In SNMPv3, there are nine possible user configurations. Each possible configuration, along with an example of each, is given below. These particular keys were generated using `defaultpassword` for the password and `9.3.149.49` as the IP address. The following command was used:

```
pwtokey –u all –p all defaultpassword 9.3.149.49
```

The following authentication and privacy keys were generated:

```
Display of 16 byte HMAC–MD5 authKey:
   18a2c7b78f3df552367383eef9db2e9f

 Display of 16 byte HMAC–MD5 localized authKey:
   a59fa9783c04bcbe00359fb1e181a4b4

 Display of 16 byte HMAC–MD5 privKey:
   18a2c7b78f3df552367383eef9db2e9f

 Display of 16 byte HMAC–MD5 localized privKey:
   a59fa9783c04bcbe00359fb1e181a4b4

 Display of 20 byte HMAC–SHA authKey:
   754ebf6ab740556be9f0930b2a2256ca40e76ef9

 Display of 20 byte HMAC–SHA localized authKey:
   cd988a098b4b627a0e8adc24b8f8cd02550463e3

 Display of 20 byte HMAC–SHA privKey:
   754ebf6ab740556be9f0930b2a2256ca40e76ef9

 Display of 16 byte HMAC–SHA localized privKey:
   cd988a098b4b627a0e8adc24b8f8cd02
```

These entries would appear in the **/etc/snmpdv3.conf** file. The following nine configurations are possible:

- Localized authentication and privacy keys using the HMAC–MD5 protocol:

  ```
  USM_USER user1 – HMAC–MD5 a59fa9783c04bcbe00359fb1e181a4b4 DES
  a59fa9783c04bcbe00359fb1e181a4b4 L – –
  ```

- Non–localized authentication and privacy keys using the HMAC–MD5 protocol:

  ```
  USM_USER user2 – HMAC–MD5 18a2c7b78f3df552367383eef9db2e9f DES
  18a2c7b78f3df552367383eef9db2e9f N  – –
  ```

- Localized authentication key using the HMAC–MD5 protocol:

  ```
  USM_USER user3 – HMAC–MD5 a59fa9783c04bcbe00359fb1e181a4b4 – – L –
  ```

- Non–localized authentication key using the HMAC–MD5 protocol:

  ```
  USM_USER user4 – HMAC–MD5 18a2c7b78f3df552367383eef9db2e9f – – N –
  ```

- Localized authentication and privacy keys using the HMAC–SHA protocol:

  ```
  USM_USER user5 – HMAC–SHA cd988a098b4b627a0e8adc24b8f8cd02550463e3 DES
  cd988a098b4b627a0e8adc24b8f8cd02 L –
  ```

- Non–localized authentication and privacy keys using the HMAC–SHA protocol:

  ```
  USM_USER user6 – HMAC–SHA 754ebf6ab740556be9f0930b2a2256ca40e76ef9 DES
  754ebf6ab740556be9f0930b2a2256ca40e76ef9 N –
  ```

- Localized authentication key using the HMAC–SHA protocol:

  ```
  USM_USER user7 – HMAC–SHA cd988a098b4b627a0e8adc24b8f8cd02550463e3 – – L
  –
  ```

- Non–localized authentication key using the HMAC–SHA protocol:

```
USM_USER user8 – HMAC–SHA 754ebf6ab740556be9f0930b2a2256ca40e76ef9 – – N
–
```

- Neither authentication nor privacy keys used (SNMPv1)

```
USM_USER user9 – none – none – – –
```

Configuring users in SNMPv3 requires configuration of both the **/etc/snmpdv3.conf** file and the **/etc/clsnmp.conf** file. For a scenario on generating user keys and editing the necessary configuration files, see Create Users in SNMPv3. In addition, see the command descriptions for the **pwtokey** and **clsnmp** commands in *AIX 5L Version 5.2 Commands Reference*, and the file formats for the **/etc/clsnmp.conf** and **/etc/snmpdv3.conf** files in *AIX 5L Version 5.2 Files Reference*. You can also refer to the sample **snmpdv3.conf** and **clsnmp.conf** configuration files located in the **/usr/samples/snmpdv3** directory.

Updating Keys

SNMPv3 offers the capability of updating user keys based on new passwords dynamically. This is done by using the **pwchange** command to generate new user keys based on an updated password, using the **clsnmp** command to dynamically update the user key in the **/etc/snmpdv3.conf** file, and editing the **/etc/clsnmp.conf** file with the new keys. During this process, the new password is never communicated between machines.

For step–by–step instructions on updating user keys, see Dynamically update authentication and privacy keys in SNMPv3 on page 1-16. In addition, refer to the **pwchange** and **clsnmp** command descriptions in *AIX 5L Version 5.2 Commands Reference* and the **/etc/clsnmp.conf** and **/etc/snmpdv3.conf** file formats in *AIX 5L Version 5.2 Files Reference*

Issuing SNMPv3 requests

The **clsnmp** command is used to send SNMP requests to SNMP agents on local or remote hosts. The requests can be SNMPv1, SNMPv2c, or SNMPv3 requests. In order to process requests, the **/etc/clsnmp.conf** file must be configured.

The **clsnmp** command can issue get, getnext, getbulk, set, walk, and findname requests. Each of these requests is briefly described below:

get	allows the user to gather data from one MIB variable
getnext	gives the next MIB variable in the MIB subtree
getbulk	gives all MIB variables from multiple MIB subtrees
set	allows the user to set a MIB variable
walk	gives all MIB variables of one subtree
findname	maps the OID to the variable name
trap	allows clsnmp to listen to traps on port 162

For detailed information on issuing clsnmp requests, see the **clsnmp** command description for in *AIX 5L Version 5.2 Commands Reference*.

Troubleshooting SNMPv3

Following are some possible problems that may be encountered.

- After upgrading from an earlier version of AIX to AIX 5.2, SNMP does not work the way it did before the migration.

  You need to migrate the community and smux entries defined in the **/etc/snmpd.conf** file to the **/etc/snmpdv3.conf** file. For information on migrating this information, see Migrate from SNMPv1 to SNMPv3 on page 1-7.

- My requests are not generating any responses.

  The most likely cause of this problem is a configuration error in either the **/etc/snmpdv3.conf** or **/etc/clsnmp.conf** files, or both. Carefully review these files to ensure that all information is entered correctly. For information on editing these files when creating new users, see Create Users in SNMPv3 on page 1-11.

- I have configured a new user using both authentication and privacy keys, but I get an error message when using this user.

  The most likely cause is that you are not running the SNMPv3 encrypted version. Follow these steps to determine what version you are running:

  1. Run `ps –e|grep snmpd`.
     - If you received no output, you probably need to start the **snmpd** daemon. Run `startsrc –s snmpd`.
     - If your output included `snmpdv1`, you are running SNMPv1. You will be able to make SNMPv1 requests when running this version.
     - If your output included `snmpdv3ne`, you are running the SNMPv3 non–encrypted version. After installing AIX 5.2, this version will be running. This version does not allow you to use privacy keys.
     - If your output included `snmpdv3e` you are running the SNMPv3 encrypted version, which is a separately–installable product. The SNMPv3 encrypted is available on the AIX Expansion Pack where allowed. The SNMPv3 encrypted version allows the use of privacy keys.
  2. Determine if the version that you are running is the intended version. If it is not, use the **snmpv3_ssw** command to change the version as follows:
     - `snmpv3_ssw –1` will switch to SNMPv1
     - `snmpv3_ssw –n` will switch to SNMPv3 non–encrypted
     - `snmpv3_ssw –e` will switch to SNMPv3 encrypted if it is installed

- After making changes to the **/etc/snmpdv3.conf** file and refreshing the daemon, my changes are not taking effect.

  After making changes to the **/etc/snmpdv3.conf** file, the SNMP daemon must be stopped and started. Refreshing the daemon will not work. Use the following procedure:

  1. Stop the SNMP daemon by running `stopsrc –s snmpd`.
  2. Start the SNMP daemon by running `startsrc –s snmpd`.

- The DPI2 subagent is started, but I cannot query any MIB variables from it?

  The most likely cause is that the `public` community is not configured in the **/etc/snmpdv3.conf** file. By default, and DPI2 subagent shipped with AIX uses the community name `public` to connect itself to the SNMP agent. The `public` community is configured in the **/etc/snmpdv3.conf** file by default. If you have removed the `public` community from the **/etc/snmpd.conf** file, add the following lines to the file:

```
VACM_GROUP group1 SNMPv1 public –
VACM_VIEW defaultView 1.3.6.1.4.1.2.2.1.1.1.0 – included –
VACM_ACCESS  group1 – – noAuthNoPriv SNMPv1  defaultView – defaultView –
COMMUNITY public    public     noAuthNoPriv 0.0.0.0     0.0.0.0         –
```

`1.3.6.1.4.1.2.2.1.1.1.0` is the OID for dpiPortForTCP.0.

- I cannot query MIB variables that are managed by the smux peer as I could before I migrated.

  Ensure that your smux entry is present in the **/etc/snmpdv3.conf** and the **/etc/snmpd.peers** files. If you configure new smux peers, ensure that they are entered in both of these files as well.

- I have implemented my own set of MIB variables, but I cannot include or exclude them from users' views.

  In the VACM_VIEW entry in the **/etc/snmpdv3.conf** file, you must specify the OID of the MIB variable rather than the MIB variable name.

- I'm not receiving traps.

  Ensure that you have configured the trap entries correctly in the **/etc/snmpdv3.conf** file. In addition, if the trap is an SNMPv3 trap, the **/etc/clsnmp.conf** file must also be configured. For instructions on configuring traps, see Create Users in SNMPv3 on page 1-11.

  In addition, make sure that the machine specified to receive traps (in the **/etc/snmpdv3.conf** file) is listening for them. You can start this process by running `clsnmp trap` on the command line.

- Why is the DPI2 server not running in the SNMPv3 environment?

  In the SNMPv3 architecture, the SNMPv3 agent itself runs the DPI2 server. See SNMPv3 Architecture on page 5-5 for more information.

SNMPv1

The information found in this section is specific to SNMPv1.

- SNMPv1 Access Policies on page 5-14
- SNMP Daemon on page 5-15
- Configuring the SNMP Daemon on page 5-16
- SNMP Daemon Processing on page 5-17
- SNMP Daemon Support for the EGP Family of MIB Variables on page 5-21
- Troubleshooting the SNMP Daemon on page 5-35

SNMPv1 Access Policies

When using SNMPv1, the **snmpd** agent uses a simple authentication scheme to determine which Simple Network Management Protocol (SNMP) manager stations can access its Management Information Base (MIB) variables. This authentication scheme involves the specification of SNMP access policies for SNMPv1. An SNMP access policy is an administrative relationship involving an association among an SNMP community, an access mode, and an MIB view.

An *SNMP community* is a group of one or more hosts and a community name. A community name is a string of octets that an SNMP manager must embed in an SNMP request packet for authentication purposes.

The *access mode* specifies the access the hosts in the community are allowed with respect to retrieving and modifying the MIB variables from a specific SNMP agent. The access mode must be one of the following: *none*, *read–only*, *read–write*, or *write–only*.

A *MIB view* defines one or more MIB subtrees that a specific SNMP community can access. The MIB view can be the entire MIB tree or a limited subset of the entire MIB tree.

When the SNMP agent receives a request, the agent verifies the community name with the requesting host IP address to determine if the requesting host is a member of the SNMP community identified by the community name. If the requesting host is a member of the SNMP community, the SNMP agent then determines if the requesting host is allowed the specified access for the specified MIB variables as defined in the access policy associated with that community. If all criteria are met, the SNMP agent attempts to honor the request. Otherwise, the SNMP agent generates an *authenticationFailure* trap or returns the appropriate error message to the requesting host.

The SNMPv1access policies for the **snmpd** agent are user–configurable and are specified in the **/etc/snmpd.conf** file. To configure the SNMP access policies for the **snmpd** agent, see the **/etc/snmpd.conf** file.

SNMP Daemon

The Simple Network Management Protocol (SNMP) daemon is a background server process that can be run on any Transmission Control Protocol/Internet Protocol (TCP/IP) workstation host. The daemon, acting as SNMP agent, receives, authenticates, and processes SNMP requests from manager applications. See Simple Network Management Protocol, How a Manager Functions, and How an Agent Functions in *AIX 5L Version 5.2 Communications Programming Concepts* for more detailed information on agent and manager functions.

Note:: The terms SNMP daemon, SNMP agent, and agent are used interchangeably.

The **snmpd** daemon requires the loopback TCP/IP interface to be active for minimal configuration. Enter the following command before starting TCP/IP:

```
ifconfig lo0 loopback up
```

Configuring the SNMP Daemon

The Simple Network Management Protocol (SNMP) daemon will attempt to bind sockets to certain well–known User Datagram Protocol (UDP) and Transmission Control Protocol (TCP) ports, which must be defined in the **/etc/services** file as follows:

```
snmp            161/udp
snmp–trap       162/udp
smux            199/tcp
```

The `snmp` service must be assigned port `161`, as required by RFC 1157. The **/etc/services** file assigns ports 161, 162, and 199 to these services. If the **/etc/services** file is being serviced off another machine, these assigned ports must be made available in the served **/etc/services** file on the server before the **SNMP** daemon can run.

The **SNMP** daemon reads the configuration file on the running SNMP version on startup and when a **refresh** command (if the **snmpd** daemon is invoked under System Resource Controller control) or **kill –1** signal is issued.

/etc/snmpd.conf file

The **/etc/snmpd.conf** configuration file specifies community names and associated access privileges and views, hosts for trap notification, logging attributes, **snmpd** –specific parameter configurations, and single multiplexer (SMUX) configurations for the **SNMP** daemon for SNMPv1. See the **/etc/snmpd.conf** file for more information.

SNMP Daemon Processing

The Simple Network Management Protocol (SNMP) daemon processes SNMP requests from manager applications. Read Simple Network Management Protocol (SNMP), How a Manager Functions, and How an Agent Functions in *AIX 5L Version 5.2 Communications Programming Concepts* for more detailed information on agent and manager functions.

Message Processing and Authentication

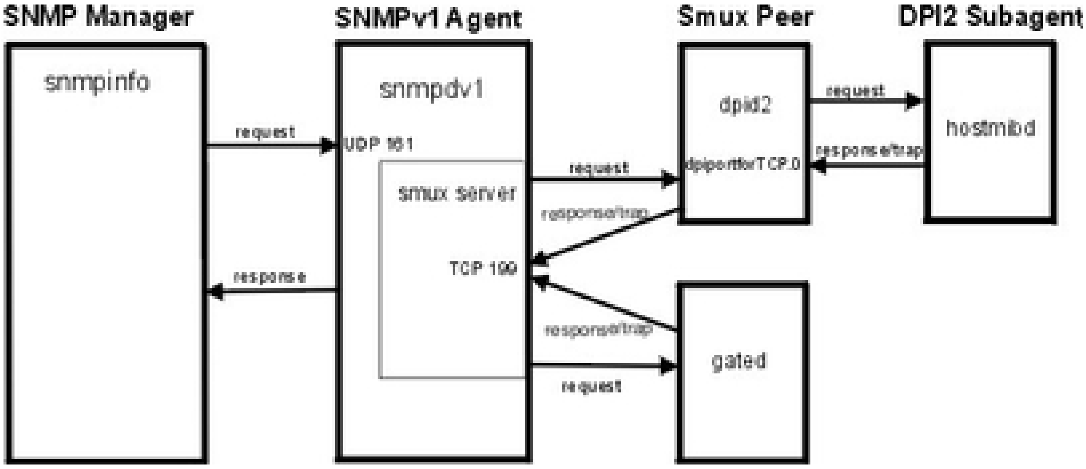
All requests, traps, and responses are transmitted in the form of ASN.1–encoded messages. A message, as defined by RFC 1157, has the following structure:

Version Community PDU

where *Version* is the SNMP version (currently version 1), *Community* is the community name, and *PDU* is the protocol data unit that contains the SNMP request, response, or trap data. A PDU is also encoded according to ASN.1 rules.

Figure 28. The primary parts of the SNMPv1 architecture

This illustration shows and example of the SNMPv1 architecture. The DPI2 subagent, smux peer, SNMP manager, and SNMP agent are shown. In addition, how they communicate with each other is shown.



The **SNMP** daemon receives and transmits all **SNMP** protocol messages through the Transmission Control Protocol/Internet Protocol (TCP/IP) User Datagram Protocol (UDP). Requests are accepted on well–known port 161. Traps are transmitted to the hosts listed in the trap entries in the **/etc/snmpd.conf** file that are listening on well–known port 162.

When a request is received, the source IP address and the community name are checked against a list containing the IP addresses, community names, permissions, and views as specified in the community and view entries in the **/etc/snmpd.conf** file. The **snmpd** agent reads this file at startup and on a **refresh** command or a **kill –1** signal. If no matching entry is found, the request is ignored. If a matching entry is found, access is allowed according to the permissions specified in the community and view entries for that IP address, community, and view name association in the **/etc/snmpd.conf** file. Both the message and the PDU must be encoded according to the ASN.1 rules.

This authentication scheme is not intended to provide full security. If the **SNMP** daemon is used only for get and get–next requests, security might not be a problem. If set requests are allowed, the set privilege can be restricted.

See the **/etc/snmpd.conf** file for further information. See Management Information Base (MIB) for further information.

Request Processing

There are three types of request PDUs that can be received by the **SNMP** daemon. The request types are defined in RFC 1157, and the PDUs all have the following format:

Request PDU format

request–ID	error–status	error–index	variable–bindings
GET	0	0	*VarBindList*
GET–NEXT	0	0	*VarBindList*
SET	0	0	*VarBindList*

The request–ID field identifies the nature of the request; the error–status field and error–index field are unused and must be set to 0 (zero); and the variable–bindings field contains a variable–length list of numeric–format instance IDs whose values are being requested. If the value of the request–ID field is SET, the variable–bindings field is a list of pairs of instance IDs and values.

Read Using the Management Information Base (MIB) Database for a discussion of the three request types.

Response Processing

Response PDUs have nearly the same format as request PDUs:

Response PDU format

request–ID	error–status	error–index	variable–bindings
GET–RESPONSE	*ErrorStatus*	*ErrorIndex*	*VarBindList*

If the request was successfully processed, the value for both the error–status and error–index field is 0 (zero), and the variable–bindings field contains a complete list of pairs of instance IDs and values.

If any instance ID in the variable–bindings field of the request PDU was not successfully processed, the SNMP agent stops processing, writes the index of the failing instance ID into the error–index field, records an error code in the error–status field, and copies the partially completed result list into the variable–bindings field.

RFC 1157 defines the following values for the error–status field:

Values for the error–status field

Value	Value	Explanation
noError	0	Processing successfully completed (error–index is 0).
tooBig	1	The size of the response PDU would exceed an implementation–defined limit (error–index is 0).
noSuchName	2	An instance ID does not exist in the relevant MIB view for GET and SET request types or has no successor in the MIB tree in the relevant MIB view for GET–NEXT requests (nonzero error–index).
badValue	3	For SET requests only, a specified value is syntactically incompatible with the type attribute of the corresponding instance ID (nonzero error–index).

readOnly	4	Not defined.
genErr	5	An implementation–defined error occurred (nonzero error– index); for example, an attempt to assign a value that exceeds implementation limits.

Trap Processing

Trap PDUs are defined by RFC 1157 to have the following format:

Trap PDU format

enterprise	**agent–address**	**generic–trap**	**specific–trap**	**time–stamp**	**variable–bindings**
Object ID	*Integer*	*Integer*	*Integer*	*TimeTicks*	*VarBindList*

The fields are used as follows:

enterprise
The object identifier assigned to the vendor implementing the agent. This is the value of the **sysObjectID** variable, and it is unique for each implementer of an SNMP agent. The value assigned to this implementation of the agent is **1.3.6.1.4.1.2.3.1.2.1.1.3**, or **risc6000snmpd.3**.

agent–address
IP address of the object generating the trap.

generic–trap
Integer, as follows:

0
coldStart

1
warmStart

2
linkDown

3
linkUp

4
authenticationFailure

5
egpNeighborLoss

6
enterpriseSpecific

specific–trap
Unused, reserved for future development.

time–stamp
Elapsed time, in hundredths of a second, from the last reinitialization of the agent to the event generating the trap.

variable–bindings
Extra information, dependent on *generic–trap* type.

The following generic–trap values indicate that certain system events have been detected:

coldStart
The agent is reinitializing. Configuration data or MIB variable values, or both, might have changed. Restart the measurement epochs.

warmStart
The agent is reinitializing but configuration data or MIB variable values have not changed. In this implementation of the SNMP agent, a *warmStart* trap is generated when the **/etc/snmpd.conf** file is reread. The configuration information in the **/etc/snmpd.conf** file is for agent configuration that has no side effects on SNMP manager databases. Measurement epochs should not be restarted.

linkDown
The agent has detected that a known communications interface has been disabled.

linkUp
The agent has detected that a known communications interface has been enabled.

authenticationFailure
A message was received that could not be authenticated.

egpNeighborLoss
An Exterior Gateway Protocol (EGP) neighbor was lost. This value is only generated when the agent is running on a host that runs the **gated** daemon using EGP.

enterpriseSpecific
Not implemented; reserved for future use.

The *linkDown* and *linkUp* traps contain a single instance ID/value pair in the variable–bindings list. The instance ID identifies the **ifIndex** of the adapter that was disabled or enabled, and the value is the **ifIndex** value. The trap for *egpNeighborLoss* also contains a binding consisting of the instance ID and value of *egpNeighAddr* for the lost neighbor.

SNMP Daemon Support for the EGP Family of MIB Variables

If the agent host is running the **gated** daemon with the Exterior Gateway Protocol (EGP) enabled, there are several Management Information Base (MIB) variables in the EGP group supported by the **gated** daemon which the **snmpd** agent can access.

The following EGP MIB variables have a single, unique instance:

egpInMsgs Number of EGP messages received without error.

egpInErrors Number of EGP messages received in error.

egpOutMsgs Total number of EGP messages transmitted by the **gated** daemon running on the agent's host.

egpOutErrors Number of EGP messages that could not be sent by the agent host **gated** daemon because of resource limitations.

egpAs Autonomous system number of the agent host **gated** daemon.

The following EGP MIB variables have an instance for each EGP peer or neighbor acquired by the agent host **gated** daemon:

egpNeighState The state of this EGP peer:

1
idle

2
acquisition

3
down

4
up

5
cease.

egpNeighAddr IP address of this EGP peer.

egpNeighAs Autonomous system number of this EGP peer. Zero (0) indicates the autonomous system number of this peer is not yet known.

egpInNeighMsgs Number of EGP messages received without error from this EGP peer.

egpNeighInErrs Number of EGP messages received in error from this EGP peer.

egpNeighOutMsgs Number of locally generated EGP messages to this EGP peer.

egpNeighOutErrs Number of locally generated EGP messages not sent to this EGP peer because of resource limitations.

egpNeighInErrMsgs Number of EGP–defined error messages received from this EGP peer

egpNeighOutErrMsgs Number of EGP–defined error messages sent to this EGP peer.

egpNeighStateUp Number of EGP state transitions to the UP state with this EGP peer.

egpNeighStateDowns Number of EGP state transitions from the UP state to any other state with this EGP peer.

egpNeighIntervalHello Interval between EGP Hello command retransmissions in hundredths of a second.

egpNeighIntervalPoll Interval between EGP poll command retransmissions in hundredths of a second.

egpNeighMode Polling mode of this EGP peer. The mode can be either active (1) or passive (2).

egpNeighEventTrigger Control variable triggers operator–initiated start and stop events to this EGP peer. This MIB variable can be set to start (1) or stop (2).

If the **gated** daemon is not running, or if the **gated** daemon is running but is not configured to communicate with the **snmpd** agent, or if the **gated** daemon is not configured for EGP, get and set requests for the values of these variables will return the *noSuchName* error response code.

The **gated** daemon configuration file, **/etc/ gated.conf**, should contain the following statement:

```
snmp      yes;
```

The **gated** daemon is internally configured to be an Simple Network Management Protocol (SNMP) single multiplexer (SMUX) protocol peer, or proxy agent, of the **snmpd** daemon. When the **gated** daemon starts up, it registers the *ipRouteTable* MIB variable tree with the **snmpd** agent. If the **gated** daemon is configured for EGP, then the **gated** daemon also registers the EGP MIB variable tree. After this registration is complete, an SNMP manager can successfully make requests to the **snmpd** agent for the *ipRouteTable* an EGP MIB variables supported by this agent host **gated** daemon. When the **gated** daemon is running, all MIB routing information is obtained using the **gated** daemon. In this case, set requests to the *ipRouteTable* are not allowed.

The SMUX communication between the **gated** daemon and the **snmpd** daemon takes place over the well–known Transmission Control Protocol (TCP) port 199. If the **gated** daemon terminates, **snmpd** immediately unregisters the trees the **gated** daemon previously registered. If the **gated** daemon is started before the **snmpd** daemon, the **gated** daemon periodically checks for the **snmpd** daemon until the SMUX association can be established.

To configure the **snmpd** agent to recognize and allow the SMUX association with the **gated** daemon client, the user must add a SMUX entry to the **/etc/ snmpd.conf** file. The client object identifier and password specified in this SMUX entry for the **gated** daemon must match those specified in the **/etc/snmpd.peers** file.

The **snmpd** agent supports set requests for the following read–write MIB I and MIB II variables:

sysContact The textual identification of the contact person for this agent host. This information includes the name of this person and how to contact this person: for example, ”Bob Smith, 555–5555, ext 5.” The value is limited to 256 characters. If, for a set request, the string for this MIB variable is greater than 256 characters, the **snmpd** agent will return the error *badValue*, and the set operation is not performed. The initial value of *sysContact* is defined in **/etc.snmp.conf**. If nothing is defined, the value is null string.

Instance	Value	Action
0	”string”	The MIB variable is set to ”string”.

sysName The host name for this agent host. Typically this is the fully qualified domain name of the node. The value is limited to 256 characters. If, for a set request, the string for this MIB variable is greater than 256 characters, the **snmpd** agent returns the error *badValue*, and the set operation is not performed.

Instance	Value	Action
0	”string”	The MIB variable is set to ”string”.

sysLocation A textual string stating the physical location of the machine on which this **snmpd** agent resides: for example, ”Austin site, building 802, lab 3C–23.” The value is limited to 256 characters. If, for a set request, the string for this MIB variable is greater than 256 characters, the **snmpd** agent returns the error *badValue*, and the set operation is not performed. The initial value of *sysLocation* is defined in **/etc/snmp.conf**. If nothing is defined, the value is null string.

Instance	Value	Action
0	”string”	The MIB variable is set to ”string”.

ifAdminStatus The desired state of an interface adapter on the agent’s host. Supported states are up and down. The state can be set to testing but such an action has no effect on the operational state of the interface.

Instance	Value	Action
f	1	The interface adapter with **ifIndex f** is enabled.

Note:: It is possible that the *ifAdminStatus* value can be set to up or down, yet the actual operational change of the interface failed. In such a case, a get request of the *ifAdminStatus* might reflect *up* while an *ifOperStatus* for that interface might reflect *down*. If such a situation occurs, the network administrator would issue another set request to set *ifAdminStatus* to up to attempt the operational change again.

atPhysAddress

The hardware address portion of an address table binding on the agent host (an entry in the Address Resolution Protocol table). This is the same MIB variable as *ipNetToMediaPhysAddress*.

Instance	Value	Action
f.1.n.n.n.n	hh:hh:hh:hh:hh:hh	For the interface with **ifIndex f,** any existing ARP table binding for IP address n.n.n.n is replaced with the binding (n.n.n.n, hh:hh:hh:hh:hh:hh). If a binding did not exist, the new binding is added. hh:hh:hh:hh:hh:hh is a twelve–hexadecimal–digit hardware address.

atN0etAddress The IP address corresponding to the hardware or physical address specified in *atPhysAddress*. This is the same MIB variable as *ipNetToMediaNetAddress*.

Instance	Value	Action
f.1.n.n.n.n	m.m.m.m	For the interface with **ifIndex f,** an existing ARP table entry for IP address n.n.n.n is replaced with IP address m.m.m.m.

ipForwarding Indicates whether this agent host is forwarding datagrams.

Instance	Value	Action
0	1	If the agent host has more than one active interface, then the TCP/IP kernel is configured to forward packets. If the agent host has only one active interface, the set request fails.
	2	The TCP/IP kernel on the agent host is configured to not forward packets.

ipDefaultTTL The default time–to–live (TTL) value inserted into IP headers of datagrams originated by the agent host.

Instance	Value	Action
0	n	The default time–to–live value used by IP protocol support is set to the integer n.

ipRouteDest The destination IP address of a route in the route table.

Instance	Value	Action
n.n.n.n	m.m.m.m	The destination route for route n.n.n.n is set to the IP address m.m.m.m.

ipRouteNextHop
The gateway by which a destination IP address can be reached from the agent host (an entry in the route table).

Instance	Value	Action
n.n.n.n	m.m.m.m	A route table entry to reach network n.n.n.n using gateway m.m.m.m is added to the route table. The host portion of the IP address n.n.n.n must be 0 to indicate a network address.

ipRouteType The state of a route table entry on the agent host (used to delete entries).

Instance	Value	Action
h.h.h.h	1	Any route to host IP address h.h.h.h is deleted.
n.n.n.n	2	Any route to host IP address n.n.n.n is deleted.

ipNetToMediaPhysAddress

The hardware address portion of an address table binding on the agent host (an entry in the ARP table). This is the same MIB variable as *atPhysAddress*.

Instance	Value	Action
f.1.n.n.n.n	hh:hh:hh:hh:hh:hh	For the interface with **ifIndex f,** any existing ARP table binding for IP address n.n.n.n is replaced with the binding (n.n.n.n, hh:hh:hh:hh:hh:hh). If a binding did not exist, the new binding is added. hh:hh:hh:hh:hh:hh is a 12–hexadecimal–digit hardware address.

ipNetToMediaNetAddress

The IP address corresponding to the hardware or physical address specified in *ipNetToMediaPhysAddress*. This is the same MIB variable as *atNetAddress*.

Instance	Value	Action
f.1.n.n.n.n	m.m.m.m	For the interface with **ifIndex f,** an existing ARP table entry for IP address n.n.n.n is replaced with IP address m.m.m.m.

ipNetToMediaType
The type of mapping from the IP address to the physical address.

Instance	Value	Action
f.1.n.n.n.n	1	For the interface with **ifIndex f,** for an existing ARP binding from IP address to physical address, the mapping type is set to 1, or other.
	2	For the interface with **ifIndex f,** for an existing ARP binding from IP address to physical address, the mapping type is set to 2, or invalid. As a side effect, the corresponding entry in the **ipNetMediaTable** is invalidated; that is, the interface is disassociated from this **ipNetToMediaTable** entry.
	3	For the interface with **ifIndex f,** for an existing ARP binding from IP address to physical address, the mapping type is set to 3, or dynamic.
	4	For the interface with **ifIndex f,** for an existing ARP binding from IP address to physical address, the mapping type is set to 4, or static.

snmpEnableAuthenTraps
Indicates whether the **snmpd** agent is configured to generate *authenticationFailure* traps.

Instance	Value	Action
0	1	The **snmpd** agent will generate authentication failure traps.
	2	The **snmpd** agent will not generate authentication failure traps.

smuxPstatus The status of an SMUX protocol peer (used to delete SMUX peers).

Instance	Value	Action
n	1	**snmpd** agent does nothing.
	2	**snmpd** agent stops communicating with SMUX peer n.

smuxTstatus The status of a SMUX MIB tree (used to delete MIB tree mounts).

Instance	Value	Action
l.m.m.m._ _ _ .p	1	**snmpd** agent does nothing.
	2	Unmounts SMUX mounting of MIB tree m.m.m... where *l* is the length of MIB tree instance and *p* is the smuxTpriority.

The following variables are the settable variables as defined in RFC 1229. The **snmpd** daemon allows the user to set these variables. The underlying device might not allow the setting of such variables. Check with each device to see what is and is not supported.

ifExtnsPromiscuous

The status of the promiscuous mode on a given device. This is used to enable and disable promiscuous mode on a given device. The **snmpd** action is final and complete. When **snmpd** is told to turn off, promiscuous mode is turned completely off regardless of the other applications on the machine.

Instance	Value	Action
n	1	Turns on the promiscuous mode for device n.
	2	Turns off the promiscuous mode for device n.

ifExtnsTestType

The test initiation variable. When this variable is set, the appropriate test is run for that device. An Object Identifier is the value of the variable. The specific value is dependent on the device type and the test that is to be run. Currently, the only defined test that **snmpd** knows to run is the testFullDuplexLoopBack test.

Instance	Value	Action
n	oid	Start the test specified by oid.

ifExtnsRcvAddrStatus

The address status variable. When this variable is set, the specified address comes into existence with the appropriate level of duration. **snmpd** only allows the setting of temporary addresses because it is not able to set device Object Data Manager (ODM) records and it is only allowed to set multicast or broadcast addresses.

Instance	Value	Action
n.m.m.m.m.m.m	1	Add the address as something other than a temporary or permanent address.
	2	Remove the address from usage.
	3	Add the address as a temporary address.
	4	Add the address as a permanent address.

The variables listed below are the settable variables as defined in RFC 1231. The **snmpd** daemon allows the user to set these variables. The underlying device might not allow the setting of such variables. You should check with each device to see what is supported.

dot5Commands

The command the token–ring device is toperform.

Instance	Value	Action
n	1	Does nothing. Returned.
	2	Tells the token–ring device to open.
	3	Tells the token–ring to reset.
	4	Tells the token–ring device to close.

dot5RingSpeed

The current ring speed or bandwidth.

Instance	Value	Action
n	1	An unknown speed.
	2	1 megabit ring speed.
	3	4 megabit ring speed.
	4	16 megabit ring speed.

dot5ActMonParticipate

The object specifies whether the device participates in the active monitor selection process.

Instance	Value	Action
n	1	Participates.
	2	Not participate.

dot5Functional

The functional mask that allows the token–ring device to specify what addresses it receives frames from.

Instance	Value	Action
n	m.m.m.m.m.m	Functional mask to be set.

The following complex timer manipulations variables are defined in the RFC as read–only but you are encouraged to make them read–write. Review the RFC to gain a full understanding of their interactions. **snmpd** allows the requestor to set them, but the device might not. Check the device driver documentation for more information. The variables are:

- dot5TimerReturnRepeat
- dot5TimerHolding
- dot5TimerQueuePDU
- dot5TimerValidTransmit
- dot5TimerNoToken
- dot5TimerActiveMon
- dot5TimerStandbyMon
- dot5TimerErrorReport

- dot5TimerBeaconTransmit
- dot5TimerBeaconReceive.

The SNMP daemon allows the user to set the following variables. The daemon uses the FDDI Station Management (SMT) 7.2 protocol standard to get the information and is determined at the microcode level. Check the microcode on the FDDI documentation to ensure that the SMT 7.2 microcode is being used.

fddimibSMTUserData

A variable holding 32 bytes of user information.

Instance	Value	Action
n	string	Stores 32 bytes of user information.

fddimibSMTConfigPolicy

The status of the configuration policies, specifically the hold policy usage.

Instance	Value	Action
n	0	Do not use the hold policy.
	1	Use the hold policy.

fddimibSMTConnectionPolicy

The status of the connection policies in the FDDI node. See RFC 1512 for more information about the specific settable values.

Instance	Value	Action
n	k	Defines the connection policies.

fddimibSMTTNotify

The timer, expressed in seconds, used in the Neighbor Notification protocol. It has a range of 2 seconds to 30 seconds, and its default value is 30 seconds.

Instance	Value	Action
n	k	Defines the timer value.

fddimibSMTStatRptPolicy

The status of the status reporting frame generation.

Instance	Value	Action
n	1	Indicates that the node generates status reporting frames for implemented events.
	2	Indicates that the node does not create status reporting frames.

fddimibSMTTraceMaxExpiration

This variable defines the maximum timer expiration value for trace.

Instance	Value	Action
n	k	Defines the maximum timer expiration in milliseconds.

fddimibSMTStationAction

This variable causes the SMT entity to take a specific action. See the RFC to get specific information about this variable.

Instance	Value	Action
n	k	Defines an action on the SMT entity. Values range from 1 to 8.

fddimibMACRequestedPaths

Defines the paths the medium access control (MAC) should be inserted.

Instance	Value	Action
n.n	k	Defines the requested path for the MAC.

fddimibMACFrameErrorThreshold

Threshold for when a MAC status report is generated. Defines the number of error that must occur before a report is generated.

Instance	Value	Action
n.n	k	Defines the number of errors that must be observed before a MAC status report is generated.

fddimibMACMAUnitdataEnable

This variable determines the value of the **MA_UNITDATA_Enable** flag in RMT. The default and initial value of this flag is true (1).

Instance	Value	Action
n.n	1	Marks the MA_UNITDATA_Enable flag true.
	2	Marks the MA_UNITDATA_Enable flag false.

fddimibMACNotCopiedThreshold

A threshold for determining when a MAC condition report is generated.

Instance	Value	Action
n.n	k	Defines the number of errors that must be observed before a MAC condition report is generated.

The following three variables are timer variables that are interactive among themselves. Before changing any of these variables, you should have a good understanding of their meaning as defined in **RFC 1512**.

- fddimibPATHTVXLowerBound
- fddimibPATHTMaxLowerBound
- fddimibPATHMaxTReq

fddimibPORTConnectionPolicies

Specifies the connection policies for the specified port.

Instance	Value	Action
n.n	k	Defines the connection policies for the specified port.

fddimibPORTRequestedPaths

This variable is a list of permitted paths where each list element defines the port permitted paths. The first octet corresponds to ‘none’, the second octet to ‘tree’, and the third octet to ‘peer’.

Instance	Value	Action
n.n	ccc	Defines the port paths.

fddimibPORTLerCutoff

The link error rate estimate at which a link connection is broken. It ranges from 10**–4 to 10**–15 and is reported as the absolute value of the base 10 logarithm (default of 7).

Instance	Value	Action
n.n	k	Defines the port LerCutoff.

fddimibPORTLerAlarm

The link error rate estimate at which a link connection generates an alarm. It ranges from 10**–4 to 10**–15 and is reported as the absolute value of the base 10 logarithm of the estimate (default is 8).

Instance	Value	Action
n.n	k	Defines the port LerAlarm.

fddimibPORTAction

This variable causes the port to take a specific action. See the RFC to get specific information about this variable.

Instance	Value	Action
n	k	Defines an action on the defined port. The values range from 1 to 6.

Note:: RFC 1213 describes all variables in the *atEntry* and *ipNetToMediaEntry* tables as read–write. Set support is implemented only for the *atEntry* variables *atPhysAddress* and *atNetAddress,* and the *ipNetToMediaEntry* variables *ipNetToMediaPhysAddress, ipNetToMediaNetAddress,* and *ipNetToMediaType.* To accept set requests that might specify the remaining unsupported attributes in these two tables, set requests for the remaining variables are accepted in *atIfIndex* and *ipNetToMediaIfIndex.* No error response is returned to the set request originator, but a subsequent get request will show that the original values are retained.

In the *ipRouteEntry* table, RFC 1213 describes all variables except *ipRouteProto* as read–write. As mentioned above, set support is implemented only for the variables *ipRouteDest*, *ipRouteNextHop*, and *ipRouteType*. To accept set requests that might specify several unsupported route attributes, set requests for the remaining variables in the *ipRouteEntry* table are accepted: *ipRouteIfIndex*, *ipRouteMetric1*, *ipRouteMetric2*, *ipRouteMetric3*, *ipRouteMetric4*, *ipRouteMetric5*, *ipRouteAge*, and *ipRouteMask*. No error response is returned to the set request originator, but a subsequent get request will show that the original values are retained. The **snmpd** daemon does not coordinate routing with the **routed** daemon. If the **gated** daemon is running and has registered the *ipRouteTable* with the **snmpd** daemon, set requests to the *ipRouteTable* are not allowed.

RFC 1229 describes settable variables that **snmpd** allows. See the previous entries for actual deviations.

Examples

The following examples use the **snmpinfo** command. It is assumed that the **snmpinfo** default community name, public, has read–write access for the respective MIB subtree.

```
snmpinfo –m set sysContact.0=”Primary contact: Bob Smith, office phone:
555–5555,
 beeper: 9–123–4567. Secondary contact: John Harris, phone: 555–1234.”
```

This command sets the value of `sysContact.0` to the specified string. If an entry for `sysContact.0` already exists, it is replaced.

```
snmpinfo –m set sysName.0=”bears.austin.ibm.com”
```

This command sets the value of `sysName.0` to the specified string. If an entry for `sysName.0` already exists, it is replaced.

```
snmpinfo –m set sysLocation.0=”Austin site, building 802, lab 3C–23,
southeast
 corner of the room.”
```

This command sets the value of `sysLocation.0` to the specified string. If an entry for `sysLocation.0` already exists, it is replaced.

```
snmpinfo –m set ifAdminStatus.2=2
```

This command disables the network interface adapter which has the ifIndex of 2. If the assigned value is 1, the interface adapter is enabled.

```
snmpinfo –m set atPhysAddress.2.1.192.100.154.2=02:60:8c:2e:c2:00
 snmpinfo –m set
ipNetToMediaPhysAddress.2.1.192.100.154.2=02:60:8c:2e:c2:00
```

These two commands change the hardware address in the ARP table entry for `192.100.154.2` to `02:60:8c:2e:c2:00`. These two commands affect the same ARP table entry. The MIB variable *atPhysAddress* is a deprecated variable and is being replaced with the MIB variable *ipNetToMediaPhysAddress*. Thus, *atPhysAddress* and *ipNetToMediaPhysAddress* access the same structure in the TCP/IP kernel ARP table.

```
snmpinfo –m set atNetAddress.2.1.192.100.154.2=192.100.154.3
 snmpinfo –m set ipNetToMediaNetAddress.2.1.192.100.154.2=192.100.154.3
```

These commands change the IP address in the ARP table entry for `192.100.154.2` to `192.100.154.3`. These two commands affect the same ARP table entry. The MIB variable *atNetAddress* is a deprecated variable and is being replaced with the MIB variable *ipNetToMediaNetAddress*. Thus, *atNetAddress* and *ipNetToMediaNetAddress* access the same structure in the TCP/IP kernel ARP table.

```
snmpinfo –m set ipForwarding.0=1
```

This command sets the TCP/IP kernel so that it can forward packets if the agent host has more than one interface that is up. If the host has only one active interface, then the set request fails and the **snmpd** agent returns the error, *badValue*.

```
snmpinfo –m set ipDefaultTTL=50
```

This command allows an IP datagram using default time–to–live (TTL) to pass through up to 50 gateways before being discarded. When each gateway processes a datagram, the gateway subtracts 1 from the time–to–live field. In addition, each gateway decrements the time–to–live field by the number of seconds the datagram waited for service at that gateway before passing the datagram on to the next destination.

```
snmpinfo –m set ipRouteDest.192.100.154.0=192.100.154.5
```

This command sets the destination IP address of the route associated with 192.100.154.0 to the IP address 192.100.154.5, assuming route 192.100.154 already existed.

```
snmpinfo –m set ipRouteNextHop.192.100.154.1=129.35.38.47
```

This command sets a route to host 192.100.154.1 using the gateway host 129.35.38.47, assuming route 192.100.154.1 already existed.

```
snmpinfo –m set ipRouteNextHop.192.100.154.0=192.100.154.7
```

This command sets a route to the class C network 192.100.154 using the gateway host 192.100.154.7, assuming route 192.100.154.0 already existed. Note that the host part of the address must be 0 to indicate a network address.

```
snmpinfo –m set ipRouteType.192.100.154.5=2
```

This command deletes any route to host 192.100.154.5.

```
snmpinfo –m set ipRouteDest.129.35.128.1=129.35.128.1
                 ipRouteType.129.35.128.1=3
              ipRouteNextHop.129.35.128.1=129.35.128.90
```

This command creates a new route from host 129.35.128.90 to 129.35.128.1 as a gateway.

```
snmpinfo –m set ipNetToMediaType.2.1.192.100.154.11=4
```

This command sets the ARP table entry for 192.100.154.11 to static.

```
snmpinfo –m set snmpEnableAuthenTraps=2
```

This command causes the **snmpd** agent on the specified host to not generate *authenticationFailure* traps.

```
snmpinfo –m set smuxPstatus.1=2
```

This command invalidates the SMUX peer 1. The result is that the connection between the **snmpd** agent and this SMUX peer is terminated.

```
snmpinfo –m set smuxTstatus.8.1.3.6.1.2.1.4.21.0=2
```

This command invalidates or removes the mounting of the SMUX tree 1.3.6.1.2.1.4.21, the *ipRoute* Table. The first number in the instance indicates the number of levels in the SMUX tree identifier. The final number in the instance indicates the smuxTpriority. In this example, there are 8 levels in the SMUX tree identifier: 1.3.6.1.2.1.4.21. The priority, 0, is the highest priority.

```
snmpinfo –m set ifExtnsPromiscuous.1=1 ifExtnsPromiscuous.2=2
```

This command turns on promiscuous mode for the first device in the interfaces table and turns off promiscuous mode for the second device in the interfaces table.

```
snmpinfo –m set ifExtnsTestType.1=testFullDuplexLoopBack
```

This command starts the testFullDuplexLoopBack test on interface 1.

```
snmpinfo –m set ifExtnsRcvAddrStatus.1.129.35.128.1.3.2=2
```

This command tells interface 1 to remove physical address 129.35.128.1.3.2 from its list of acceptable addresses.

```
snmpinfo –m set dot5Commands.1=2
```

This command tells the first interface to do an open.

```
snmpinfo –m set dot5RingSpeed.1=2
```

This command tells the first interface to set it ring speed to 1 megabit.

```
snmpinfo –m set dot5ActMonParticipate.1=1
```

This command tells the first interface to participate in the active monitor selection process.

```
snmpinfo –m set dot5Functional.1=255.255.255.255.255.255
```

This command sets the functional address mask to allow everything.

```
snmpinfo –m set fddimibSMTUserData.1=”Greg’s Data”
```

This command sets the user data on the first SMT entity to ”Greg’s Data”.

```
snmpinfo –m set fddimibMACFrameErrorThreshold.1.1=345
```

This command sets the threshold for frame errors to 345 on the first MAC of the first SMT entity.

Note:: All of the variables described previously fall into one of the listed methods used to set the variable.

See Address Resolution Protocol and Internet Addresses for more information on protocols and Internet addresses.

Troubleshooting the SNMP Daemon

If the **snmpd** agent is not behaving as expected, the following are some hints to help determine and correct the problem. It is strongly recommended that you start up the **snmpd** agent with some type of logging. If invoking the **snmpd** daemon causes problems, it is strongly recommended that the **syslogd** daemon be set up for logging at the daemon facility and DEBUG severity level. See the **snmpd** command and the **snmpd.conf** file for more information on **snmpd** logging.

Daemon Termination Problem

If the **snmpd** daemon terminates as soon as it is invoked, the following are possible reasons for failure and probable solutions:

- The reason the **snmpd** daemon terminated will be logged in the **snmpd** log file or the configured **syslogd** log file. Check the log file to see the **FATAL** error message.

  Solution: Correct the problem and restart the **snmpd** daemon.

- The **snmpd** command line usage was incorrect. If the **snmpd** command was invoked without the System Resource Controller (SRC), the required usage statement is echoed to the screen. If the **snmpd** daemon was invoked under SRC control, the usage message is not echoed to the screen. Check the log file to see the usage message.

  Solution: Invoke the **snmpd** command with the correct usage statement.

- The **snmpd** daemon must be invoked by the root user.

  Solution: Switch to the root user and restart the **snmpd** daemon.

- The **snmpd.conf** file must be owned by the root user. The **snmpd** agent verifies the ownership of the configuration file. If the file is not owned by the root user, the **snmpd** agent terminates with a fatal error.

  Solution: Make sure you are the root user, change the ownership of the configuration file to the root user, and restart the **snmpd** daemon.

- The **snmpd.conf** file must exist. If the **–c** flag is not specified in the configuration file on the **snmpd** command line, the **/etc/snmpd.conf** file does not exist. If the **/etc/snmpd.conf** file is accidentally removed, reinstall the **bos.net.tcp.client** image or else reconstruct the file with the appropriate configuration entries as defined in the **snmpd.conf** file management page. If the configuration file is specified with the **–c** flag on the **snmpd** command line, make sure that the file exists and that the file is owned by the root user. The full path and file name of the configuration file must be specified or else the default **/etc/snmpd.conf** file will be used.

  Solution: Make sure the specified configuration file exists and that this file is owned by the root user. Restart the **snmpd** daemon.

- The **udp port 161** is already bound. Make sure that the **snmpd** daemon is not already running. Issue the **ps –eaf | grep snmpd** command to determine if an **snmpd** daemon process is already executing. Only one **snmpd** agent can bind to **udp port 161**.

  Solution: Either kill the existing **snmpd** agent or do not try to start up another **snmpd** daemon process.

Daemon Failure Problem

If the **snmpd** daemon fails when you issue a **refresh** or a **kill –1** signal, the following are possible reasons for failure and probable solutions:

- The reason the **snmpd** daemon terminated is logged in the **snmpd** log file or the configured **syslogd** log file. Check the log file to see the FATAL error message.

  Solution: Correct the problem and restart the **snmpd** daemon.

- Make sure that the complete path and file name of the configuration file is specified when the **snmpd** daemon is invoked. The **snmpd** daemon forks and changes to the root directory at invocation. If the complete path name of the configuration file is not specified, the **snmpd** agent cannot find the file on a refresh. This is a fatal error and will cause the **snmpd** agent to terminate.

  Solution: Specify the complete path and file name of the **snmpd** configuration file. Make sure the configuration file is owned by the root user. Restart the **snmpd** daemon.

- Make sure that the **snmpd** configuration file still exists. The file may have been accidentally removed after the **snmpd** agent was invoked. If the **snmpd** agent cannot open the configuration file, the **snmpd** agent terminates.

  Solution: Recreate the **snmpd** configuration file, make sure the configuration file is owned by the root user, and restart the **snmpd** daemon.

MIB Variable Access Problem

If Management Information Base (MIB) variables cannot be accessed from the **snmpd** agent; if the **snmpd** agent is running, but the Simple Network Management Protocol (SNMP) manager application times out waiting for a response from the **snmpd** agent, try the following:

- Check the network configuration of the host on which the **snmpd** agent is running using the **netstat –in** command. Verify the lo0, loopback, device is up. If the device is down, an * (asterisk) displays to the left of the lo0. The lo0 must be up for the **snmpd** agent to service requests.

  Solution: Issue the following command to start up the loopback interface:

```
ifconfig lo0 inet up
```

- Verify that the **snmpd** daemon has a route to the host where the requests are issued.

  Solution: On the host where the **snmpd** daemon is running, add a route to the host where the **route add** command is issued. See the **route** command for more information.

- Check to see that the host name and the host IP address are the same value.

  Solution: Reset the hostname to correspond to the host IP address.

- Check to see that *localhost* is defined to be the lo0 IP address.

  Solution: Define *localhost* to be the same address used by the lo0 IP address (usually `127.0.0.1`).

MIB Variable Access in Community Entry Problem

If a community entry is specified in the configuration file with a MIB view name, but MIB variables cannot be accessed, check the following:

- Make sure that you have correctly specified the community entry. If you have specified a view name in the community entry, all fields in the community are absolutely required.

  Solution: Specify all fields in the community entry in the configuration file. Refresh the **snmpd** agent and try your request again.

- Make sure the access mode in the community entry corresponds with your request type. If you are issuing a **get** or **get–next** request, make sure that the community has read–only or read–write permission. If you are issuing a **set** request, make sure that the community has read–write permission.

Solution: Specify the correct access mode in the community entry. Refresh the **snmpd** agent and try your request again.

- Make sure that a view entry for the specified view name is specified in the community entry in the configuration file. If there is a specified view name in the community entry, but there is no corresponding view entry, the **snmpd** agent does not allow access for that community. A view entry is absolutely required for a view name specified in a community entry in the configuration file.

  Solution: Specify a view entry for the view name specified in the community entry. Refresh the **snmpd** agent and try your request again.

- If `iso` is specified as the MIB subtree for the view entry, verify that `iso.3` is specified. The instance of `3` is required for the **snmpd** agent to access the `org` portion of the `iso` tree.

  Solution: Specify the MIB subtree as `iso.3` in the view entry. Refresh the **snmpd** agent and try your request again.

- Check the *IP address* and *network mask* in the community entry. Verify that the host issuing the SNMP request is included in the community being specified with the community name.

  Solution: Change the *IP address* and *network mask* fields in the community entry in the configuration file to include the host that is issuing the SNMP request.

No Response from Agent Problem

If the *IP address* in the community is specified as 0.0.0.0, but there is no response from the **snmpd** agent, try the following:

- Check the *network mask* field in the community entry. For general access to this community name, the *network mask* must be **0.0.0.0**. If the *network mask* is specified to be **255.255.255.255**, the **snmpd** agent is configured to not allow any requests with the specified community name.

  Solution: Specify the *network mask* in the community entry to 0.0.0.0. Refresh the **snmpd** agent and try the request again.

- Make sure the access mode in the community entry corresponds with the request type. When issuing a **get** or **get–next** request, make sure that the community has read–only or read–write permission. If you are issuing a **set** request, make sure that the community has read–write permission.

  Solution: specify the correct access mode in the community entry. Refresh the **snmpd** agent and try your request again.

noSuchName Problem

If, in attempting to set an MIB variable that the **snmpd** agent is supposed to support, a `noSuchName` error message is returned, the following might be the reason:

The set request issued did not include a community name for a valid community with write access. The SNMP protocol dictates that a set request with a community with inappropriate access privileges be answered with the `noSuchName` error message.

Solution: Issue the set request with a community name for a community that has write privileges and includes the host from which the set request is issued.

Chapter 6. Network File System and SMBFS

This chapter provides information on the Network File System (NFS), a mechanism for storing files on a network. The following topics are discussed:

For information on NFS security, see Network File System (NFS) Security in *AIX 5L Version 5.2 Security Guide*.

Network File System Overview

The Network File System (NFS) is a distributed file system that allows users to access files and directories located on remote computers and treat those files and directories as if they were local. For example, users can use operating system commands to create, remove, read, write, and set file attributes for remote files and directories.

The NFS software package includes commands and daemons for NFS, Network Information Service (NIS), and other services. Although NFS and NIS are installed together as one package, each is independent and each is configured and administered individually. See *AIX 5L Version 5.2* Network Information Service *(NIS and NIS+) Guide* for details on NIS and NIS+.

This operating system supports the latest NFS protocol update, NFS Version 3, and provides an NFS Version 2 client and server. The operating system is, therefore, backward compatible with an existing install base of NFS clients and servers.

The following topics are discussed in this section:

- NFS Services on page 6-2
- NFS Access Control Lists (ACL) Support on page 6-3
- Cache File System (CacheFS) Support on page 6-4
- NFS Mapped File Support on page 6-5
- Three Types of Mounts on page 6-6
- NFS Mounting Process on page 6-6
- /etc/exports File on page 6-7
- /etc/xtab File on page 6-7
- Implementation of NFS on page 6-7
- Controlling NFS on page 6-8

NFS Services

NFS provides its services through a client–server relationship. The computers that make their *file systems*, or *directories*, and other resources available for remote access are called *servers.* The act of making file systems available is called *exporting*. The computers and their processes that use server resources are considered *clients*. Once a client *mounts* a file system that a server exports, the client can access the individual server files (access to exported directories can be restricted to specific clients).

The major services provided by NFS are:

Mount service	From the **/usr/sbin/rpc.mountd** daemon on the server and the **/usr/sbin/mount** command on the client.
Remote File access	From the **/usr/sbin/nfsd** daemon on the server and the **/usr/sbin/biod** daemon on the client.
Remote execution service	From the **/usr/sbin/rpc.rexd** daemon on the server and the **/usr/bin/on** command on the client.
Remote System Statistics service	From the **/usr/sbin/rpc.rstatd** daemon on the server and the **/usr/bin/rup** command on the client.

Remote User Listing service
From the **/usr/lib/netsvc/rusers/rpc.rusersd** daemon on the server and the **/usr/bin/rusers** command on the client.

Boot Parameters service
Provides startup parameters to Sun Operating System diskless clients from the **/usr/sbin/rpc.bootparamd** daemon on the server.

Remote Wall service
From the **/usr/lib/netsvc/rwall/rpc.rwalld** daemon on the server and the **/usr/sbin/rwall** command on the client.

Spray service
Sends a one–way stream of Remote Procedure Call (RPC) packets from the **/usr/lib/netsvc/spray/rpc.sprayd** daemon on the server and the **/usr/sbin/spray** command on the client.

PC authentication service
Provides a user authentication service for PC–NFS from the **/usr/sbin/rpc.pcnfsd** daemon on the server.

Note:: A computer can be both an NFS server and an NFS client simultaneously.

An NFS server is *stateless* meaning that an NFS server does not have to remember any transaction information about its clients. NFS transactions are atomic. Single NFS transaction corresponds to a single, complete file operation. NFS requires that the client remember any information needed for later NFS use.

NFS Access Control Lists (ACL) Support

Although NFS supports access control lists (ACLs), they are no longer used as the default. To use access control lists with NFS, use the **acl** option with the NFS **–o** flag, as shown in the following example:

```
mount –o acl
```

This support is handled by an RPC program that exchanges information about ACLs between clients and servers. The ACL support does not change the NFS protocol specification; it is a separate function.

The operating system adds ACLs to the regular file system. Since the normal NFS protocol does not support ACLs, they cannot be seen by normal NFS clients and unexpected behavior can result. A user on an NFS client might presume access to a file after looking at the permission bits, but the permissions could have been altered by the ACL associated with the file. Permissions on a server are enforced by the server according to the ACL on the server. Therefore, a user on the client machine could receive a permissions error.

When a client first attempts to access a remote mounted file system, it attempts to contact the ACL RPC program on the server.

If the server is a AIX 3.2 server, the client consults the ACL associated with a file before granting access to the program on the client. This provides the expected behavior on the client when the request is sent over to the server. In addition, the **aclget**, **aclput**, and **alcedit** commands can be used on the client to manipulate ACLs.

Cache File System (CacheFS) Support

The Cache File System (CacheFS) is a general–purpose file system caching mechanism that improves NFS server performance and scalability by reducing server and network load. Designed as a layered file system, CacheFS provides the ability to cache one file system on another. In an NFS environment, CacheFS increases the client–per–server ratio, reduces server and network loads and improves performance for clients on slow links, such as Point–to–Point Protocol (PPP).

A cache is created on the client machine so file systems specified to be mounted in the cache can be accessed locally instead of across the network. Files are placed in the cache when a user first requests access to them. The cache does not get filled until the user requests access to a file or files. Initial file requests may seem slow, but subsequent uses of the same files are faster. Notes:

1. You cannot cache the **/** (root) or **/usr** file systems.
2. You can mount only file systems that are shared. (See the **exportfs** command.)
3. There is no performance gain in caching a local Journaled File System (JFS) disk file system.
4. You must have root or system authority to do the tasks in the following table.

CacheFS tasks

Task	**SMIT fast path**	**Command or file**	Web-based System Manager Management Environment
Set up a cache	**cachefs_admin_create**	**cfsadmin –c** *MountDirectoryName* [1].	Software —> **File Systems** —> **Cached File Systems** —> **New Cached File System**
Specifying Files for Mounting	**cachefs_mount**	**mount –F cachefs –o backfstype=** *FileSysType,* **cachedir=** *CacheDirectory* [, *options*] *BackFileSystem MountDirectoryName* [2] or edit **/etc/filesystems**.	Software —> **File Systems** —> **Overview and Tasks** —> **Mount a File System**.
Modify the Cache	**cachefs_admin_change**	remove the cache, then recreate it using appropriate **mount** command options.	Software —> **File Systems** —> **Cached File Systems** —> **Selected** —> **Delete**. Proceed by setting up a cache as previously shown in the first row of this table.

Display Cache Information	**cachefs_admin_change**	**cfsadmin –l** *MountDirectoryName.*	Software —> **File Systems** —> **Cached File Systems** —> **Selected** —> **Properties**.
Remove a Cache	**cachefs_admin_remove**	1. Unmount the file system: **umount** *MountDirectoryName* 2. Determine the cache ID: **cfsadmin –l** *MountDirectoryName* 3. Delete the file system: **cfsadmin –d** *CacheID CacheDirectory*	Software —> **File Systems** —> **Cached File Systems** —> **Selected** —> **Delete**.
Check File System Integrity	**cachefs_admin_check**	**fsck_cachefs** *CacheDirectory* [3].	Software —> **File Systems** —> **Cached File Systems** —> **Selected** —> **Check integrity of Cache**.

Notes:

1. After you have created the cache, do not perform any operations within the cache directory (**cachedir**) itself. This causes conflicts within the CacheFS software.
2. If you use the **mount** command option to specify files for mounting, the command must be reissued each time the system is restarted.
3. Use the **–m** or **–o** options of the **fsck_cachefs** command to check the file systems without making any repairs.

NFS Mapped File Support

NFS mapped file support allows programs on a client to access a file as though it were in memory. By using the **shmat** subroutine, users can map areas of a file into their address space. As a program reads and writes into this region of memory, the file is read into memory from the server or updated as needed on the server.

Mapping files over NFS is limited in three ways:

- Files do not share information well between clients.
- Changes to a file on one client using a mapped file are not seen on another client.
- Locking and unlocking regions of a file is not an effective way to coordinate data between clients.

If an NFS file is to be used for data sharing between programs on different clients, use record locking and the regular **read** and **write** subroutines.

Multiple programs on the same client can share data effectively using a mapped file. Advisory record locking can coordinate updates to the file on the client, provided that the entire file is locked. Multiple clients can share data–using mapped files only if the data never changes, as in a static database.

Three Types of Mounts

There are three types of NFS mounts:

1. Predefined
2. Explicit
3. Automatic.

.

Predefined mounts are specified in the **/etc/filesystems** file. Each stanza (or entry) in this file defines the characteristics of a mount. Data such as the host name, remote path, local path, and any mount options are listed in this stanza. Predefined mounts are used when certain mounts are always required for proper operation of a client.

Explicit mounts serve the needs of the root user. Explicit mounts are usually done for short periods of time when there is a requirement for occasional unplanned mounts. Explicit mounts can also be used if a mount is required for special tasks and that mount is not generally available on the NFS client. These mounts are usually fully qualified on the command line by using the **mount** command with all needed information. Explicit mounts do not require updating the **/etc/filesystems** file. File systems mounted explicitly remain mounted unless explicitly unmounted with the **umount** command or until the system is restarted.

Automatic mounts are controlled by the **automount** command, which causes the **AutoFS** kernel extension to monitor specified directories for activity. If a program or user attempts to access a directory that is not currently mounted, then **AutoFS** intercepts the request, arranges for the mount of the file system, then services the request.

NFS Mounting Process

Clients access files on the server by first mounting server exported directories. When a client mounts a directory, it does not make a copy of that directory. Rather, the mounting process uses a series of remote procedure calls to enable a client to transparently access the directories on the server. The following describes the mounting process:

1. When the server starts, the **/etc/rc.nfs** script runs the **exportfs** command, which reads the server **/etc/exports** file, and then tells the kernel which directories are to be exported and what access restrictions they require.
2. The **rpc.mountd** daemon and several **nfsd** daemons (8, by default) are then started by the **/etc/rc.nfs** script.
3. When the client starts, the **/etc/rc.nfs** script starts several **biod** daemons (8, by default), which forward client mount requests to the appropriate server.
4. Then the **/etc/rc.nfs** script executes the **mount** command, which reads the file systems listed in the **/etc/filesystems** file.
5. The **mount** command locates one or more servers that export the information the client wants and sets up communication between itself and that server. This process is called *binding.*
6. The **mount** command then requests that one or more servers allow the client to access the directories in the client **/etc/filesystems** file.
7. The server **rpc.mountd** daemon receives the client mount requests and either grants or denies them. If the requested directory is available to that client, the **rpc.mountd** daemon sends the client kernel an identifier called a *file handle*.
8. The client kernel then ties the file handle to the mount point (a directory) by recording certain information in a *mount record*.

Once the file system is mounted, the client can perform file operations. When the client does a file operation, the **biod** daemon sends the file handle to the server, where the file is read by one of the **nfsd** daemons to process the file request. Assuming the client has access to perform the requested file operation, the **nfsd** daemon returns the necessary information to the client **biod** daemon.

/etc/exports File

The **/etc/exports** file indicates all directories that a server exports to its clients. Each line in the file specifies a single directory. The server automatically exports the listed directories each time the NFS server is started. These exported directories can then be mounted by clients. The syntax of a line in the **/etc/exports** file is:

```
directory     –options[,option]
```

The `directory` is the full path name of the directory. Options can designate a simple flag such as `ro` or a list of host names. See the specific documentation of the **/etc/exports** file and the **exportfs** command for a complete list of options and their descriptions. The **/etc/rc.nfs** script does not start the **nfsd** daemons or the **rpc.mountd** daemon if the **/etc/exports** file does not exist.

The following example illustrates entries from an **/etc/exports** file:

```
/usr/games    –ro,access=ballet:jazz:tap
 /home          –root=ballet,access=ballet
 /var/tmp
 /usr/lib       –access=clients
```

The first entry in this example specifies that the `/usr/games` directory can be mounted by the systems named `ballet`, `jazz`, and `tap`. These systems can read data and run programs from the directory, but they cannot write in the directory.

The second entry in this example specifies that the `/home` directory can be mounted by the system `ballet` and that root access is allowed for the directory.

The third entry in this example specifies that any client can mount the `/var/tmp` directory. (Notice the absence of an access list.)

The fourth entry in this example specifies an access list designated by the netgroup `clients`. In other words, these machines designated as belonging to the netgroup `clients` can mount the `/usr/lib` directory from this server. (A *netgroup* is a network–wide group allowed access to certain network resources for security or organizational purposes. Netgroups are controlled by using NIS or NIS+. For more information, see *AIX 5L Version 5.2* Network Information Service *(NIS and NIS+) Guide*.)

/etc/xtab File

The **/etc/xtab** file has a format identical to the **/etc/exports** file and lists the currently exported directories. Whenever the **exportfs** command is run, the **/etc/xtab** file changes. This allows you to export a directory temporarily without having to change the **/etc/exports** file. If the temporarily exported directory is unexported, the directory is removed from the **/etc/xtab** file.

Note:: The **/etc/xtab** file is updated automatically, and is not to be edited.

Implementation of NFS

NFS is implemented on a wide variety of machine types, operating systems, and network architectures. NFS achieves this independence using the *Remote Procedure Call* (RPC) protocol.

Remote Procedure Call (RPC) Protocol

RPC is a library of procedures. The procedures allow one process (the client process) to direct another process (the server process) to run procedure calls as if the client process had run the calls in its own address space. Because the client and the server are two separate processes, they need not exist on the same physical system (although they can).

NFS is implemented as a set of RPC calls in which the server services certain types of calls made by the client. The client makes such calls based on the file system operations that are done by the client process. NFS, in this sense, is an RPC application.

Because the server and client processes can reside on two different physical systems which may have completely different architectures, RPC must address the possibility that the two systems might not represent data in the same way. For this reason, RPC uses data types defined by the eXternal Data Representation (XDR) protocol.

eXternal Data Representation (XDR) Protocol

XDR is the specification for a standard representation of various data types. By using a standard data type representation, a program can be confident that it is interpreting data correctly, even if the source of the data is a machine with a completely different architecture.

In practice, most programs do not use XDR internally. Rather, they use the data type representation specific to the architecture of the computer on which the program is running. When the program needs to communicate with another program, it converts its data into XDR format before sending the data. Conversely, when it receives data, it converts the data from XDR format into its own specific data type representation.

The portmap Daemon

Each RPC application has associated with it a program number and a version number. These numbers are used to communicate with a server application on a system. When making a request from a server, the client needs to know what port number the server is accepting requests on. This port number is associated with the User Datagram Protocol (UDP) or Transmission Control Protocol (TCP) that is being used by the service. The client knows the program number, the version number, and the system name or host name where the service resides. The client needs a way to map the program number and version number pair to the port number of the server application. This is done with the help of the **portmap** daemon.

The **portmap** daemon runs on the same system as the NFS application. When the server starts running, it registers with the **portmap** daemon. As a function of this registration, the server supplies its program number, version number, and UDP or TCP port number. The **portmap** daemon keeps a table of server applications. When the client tries to make a request of the server, it first contacts the **portmap** daemon to find out which port the server is using. The **portmap** daemon responds to the client with the port of the server that the client is requesting. Upon receipt of the port number, the client is able to make all of its future requests directly to the server application.

Controlling NFS

The NFS, NIS, and NIS+ daemons are controlled by the System Resource Controller (SRC). This means you must use SRC commands such as **startsrc**, **stopsrc**, and **lssrc** to start, stop, and check the status of the NFS, NIS, and NIS+ daemons.

Some NFS daemons are not controlled by the SRC: specifically, **rpc.rexd**, **rpc.rusersd**, **rpc.rwalld**, and **rpc.rsprayd**. These daemons are started and stopped by the **inetd** daemon.

The following table lists the SRC–controlled daemons and their subsystem names.

Daemons and their subsystems

File path	Subsystem name	Group name
/usr/sbin/nfsd	**nfsd**	nfs
/usr/sbin/biod	**biod**	nfs
/usr/sbin/rpc.lockd	**rpc.lockd**	nfs
/usr/sbin/rpc.statd	**rpc.statd**	nfs
/usr/sbin/rpc.mountd	**rpc.mountd**	nfs
/usr/lib/netsvc/yp/ypserv	**ypserv**	yp
/usr/lib/netsvc/yp/ypbind	**ypbind**	yp
/usr/lib/netsvc/rpc.yppasswdd	**yppasswdd**	yp
/usr/lib/netsvc/rpc.ypupdated	**ypupdated**	yp
/usr/sbin/keyserv	**keyserv**	keyserv
/usr/sbin/portmap	**portmap**	portmap

NIS+ daemons are described in *AIX 5L Version 5.2* Network Information Service *(NIS and NIS+) Guide*. Each of these daemons can be specified to the SRC commands by using their subsystem name or the appropriate group name. These daemons support neither the long–listing facility of SRC nor the SRC trace commands.

For more information on using the SRC, see System Resource Controller Overview in *AIX 5L Version 5.2 System Management Concepts: Operating System and Devices*.

Change the Number of biod and nfsd Daemons

Use the **chnfs** command to change the number of **biod** or **nfsd** daemons running on the system. For example, to set the number of **nfsd** daemons to 10 and the number **biod** daemons to 4, run the command:

```
chnfs –n 10 –b 4
```

This command temporarily stops the daemons currently running on the system, modifies the SRC database code to reflect the new number, and restarts the daemons.

Note:: In this implementation of NFS, the number of **biod** daemons are controllable only per mount point using the **biod –o** option. Specification using **chnfs** is retained for compatibility purposes only and has no real effect on the number of threads performing I/O.

Change Command Line Arguments for Daemons Controlled by SRC

Many NFS, NIS, and NIS+ daemons have command–line arguments that can be specified when the daemon is started. Since these daemons are not started directly from the command line, you must update the SRC database so that the daemons can be started correctly. To do this, use the **chssys** command. The **chssys** command has the format:

```
chssys –s   Daemon   –a ′   NewParameter  ′
```

For example:

```
chssys –s nfsd –a ′10′
```

changes the **nfsd** subsystem so that when the daemon is started, the command line looks like `nfsd 10`. The changes made by the **chssys** command do not take effect until the subsystem has been stopped and restarted.

Start the NFS Daemons at System Startup

The NFS daemons, by default, are not started during installation. When installed, all of the files are placed on the system, but the steps to activate NFS are not taken. You can start the NFS daemons at system startup through:

- The Web–based System Manager, **wsm**
- The SMIT fast path, **smit mknfs**
- The **mknfs** command.

All of these methods place an entry in the **inittab** file so that the **/etc/rc.nfs** script is run each time the system restarts. This script, in turn, starts all NFS daemons required for a particular system.

Start the NFS Daemons

The file size limit for files located on an NFS server is taken from the process environment when **nfsd** is started. To use a specific value, edit the **/etc/rc.nfs** file. Using the **ulimit** command with the desired limit before the **startsrc** command for **nfsd**.

The NFS daemons can be started individually or all at once. To start NFS daemons individually, run:

```
startsrc –s   Daemon
```

where `Daemon` is anyone of the SRC–controlled daemons. For example, to start the **nfsd** daemons, run:

```
startsrc –s nfsd
```

To start all of the NFS daemons, run:

```
startsrc –g nfs
```

Note:: If the **/etc/exports** file does not exist, the **nfsd** and the **rpc.mountd** daemons will not be started. You can create an empty **/etc/exports** file by running the command **touch /etc/exports**. This will allow the **nfsd** and the **rpc.mountd** daemons to start, although no file systems will be exported.

Stop the NFS Daemons

The NFS daemons can be stopped individually or all at once. To stop NFS daemons individually, run:

```
stopsrc –s   Daemon
```

where `Daemon` is anyone of the SRC–controlled daemons. For example, to stop the **rpc.lockd** daemon, run:

```
stopsrc –s rpc.lockd
```

To stop all NFS daemons at once, run:

```
stopsrc –g nfs
```

Get the Current Status of the NFS Daemons

You can get the current status of the NFS daemons individually or all at once. To get the current status of the NFS daemons individually, run:

```
lssrc –s   Daemon
```

where `Daemon` is any one of the SRC–controlled daemons. For example, to get the current status of the **rpc.lockd** daemon, run:

```
lssrc –s rpc.lockd
```

To get the current status of all NFS daemons at once, run:

```
lssrc –a
```

NFS Installation and Configuration

For information on installing the Network File System (NFS), see the *AIX 5L Version 5.2 Installation Guide and Reference*.

Checklist for Configuring NFS

Once the NFS software is installed on your systems, you are ready to configure NFS.

1. Determine which systems in the network are to be servers and which are to be clients (a system can be configured as both a server and a client).
2. For each system (whether client or server), follow the instructions in Start the NFS Daemons at System Startup.
3. For each NFS server, follow the instructions in Configuring an NFS Server.
4. For each NFS client, follow the instructions in Configuring an NFS Client.
5. If you want personal computers on your network to have access to your NFS servers (beyond being able to mount file systems), configure PC–NFS by following the instructions in PC–NFS.

Configuring an NFS Server

To configure an NFS server:

1. Start NFS using the instructions in Start the NFS Daemons on page 6-10.
2. Create the **/etc/exports** file.

Configuring an NFS Client

1. Verify that NFS is the default remote file system. (If this is not done, specify the **–v nfs** flag when using the **mount** command.) Using a text editor, open the **/etc/vfs** file and search for the following entry:

```
#%defaultvfs jfs nfs
 #nfs 2 /sbin/helpers/nfsmnthelp none remote
```

   Delete any lines that begin with a pound sign (#).

2. Start NFS using the instructions in Start the NFS Daemons.
3. Establish the local mount point using the **mkdir** command. For NFS to complete a mount successfully, a directory that acts as the mount point (or place holder) of an NFS mount must be present. This directory should be empty. This mount point can be created like any other directory, and no special attributes are needed.

   Note:: With one exception, the mount points for all NFS mounts must exist on your system before mounting a file system. If the **automount** daemon is used, it might not be necessary to create mount points. See the **automount** documentation for details.

4. Establish and mount the predefined mounts by following the instructions in Establishing Predefined NFS Mounts.

Exporting an NFS File System

You can export an NFS file system using the Web-based System Manager Network application, or you can use one of the following procedures.

- To export an NFS file system using the System Management Interface Tool (SMIT):
  1. Verify that NFS is already running by typing the command `lssrc –g nfs`**.** The output should indicate that the **nfsd** and the **rpc.mountd** daemons are active. If they are not, start NFS using the instructions in Start the NFS Daemons on page 6-10 .
  2. At a command line, type the following and press **Enter**:

     ```
     smit mknfsexp
     ```
  3. Specify appropriate values in the PATHNAME of directory to export, MODE to export directory, and EXPORT directory now, system restart or both fields.
  4. Specify any other optional characteristics you want, or accept the default values by leaving the remaining fields as they are.
  5. When you have finished making your changes, SMIT updates the **/etc/exports** file. If the **/etc/exports** file does not exist, it is created.
  6. Repeat steps 3 through 5 for each directory you want to export.
- To export an NFS file system using a text editor:
  1. Open the **/etc/exports** file with your favorite text editor.
  2. Create an entry for each directory to be exported using the full path name of the directory. List each directory to be exported starting in the left margin. No directory should include any other directory that is already exported. See the **/etc/exports** file documentation for a description of the full syntax for entries in the **/etc/exports** file.
  3. Save and close the **/etc/exports** file.
  4. If NFS is running, type the following comand and press **Enter**:

     ```
     /usr/sbin/exportfs –a
     ```

     The **–a** option tells the **exportfs** command to send all information in the **/etc/exports** file to the kernel. If NFS is not running, start NFS using the instructions in Start the NFS Daemons on page 6-10.
- To temporarily export an NFS file system (without changing the **/etc/exports** file), type the following command and press **Enter**:

  ```
  exportfs –i /   dirname
  ```

  where `dirname` is the name of the file system you want to export. The **exportfs –i** command specifies that the **/etc/exports** file is not to be checked for the specified directory, and all options are taken directly from the command line.

Unexporting an NFS File System

You can unexport an NFS directory using the Web-based System Manager Network application, or you can use one of the following procedures.

- To unexport an NFS directory using SMIT:
  1. Type the following at a command prompt and press **Enter**:

     ```
     smit rmnfsexp
     ```
  2. Enter the appropriate path name in the PATHNAME of exported directory to be removed field.

     The directory is now removed from the **/etc/exports** file and is unexported.
- To unexport an NFS directory by using a text editor:
  1. Open the **/etc/exports** file with your favorite text editor.

2. Find the entry for the directory you wish to unexport, and the delete that line.
3. Save and close the **/etc/exports** file.
4. If NFS is currently running, enter:

   ```
   exportfs –u   dirname
   ```

   where `dirname` is the full path name of the directory you just deleted from the **/etc/exports** file.

Changing an Exported File System

Change an exported NFS file system using the Web-based System Manager Network application, or use one of the following procedures.

- To change an exported NFS file system using SMIT:
  1. To unexport the file system, **Enter**:

     ```
     exportfs –u /   dirname
     ```

     where `dirname` is the name of the file system you want to change.
  2. **Enter**:

     ```
     smit chnfsexp
     ```
  3. Enter the appropriate path name in the PATHNAME of exported directory field.
  4. Make whatever changes you want.
  5. Exit SMIT.
  6. Re–export the file system by entering:

     ```
     exportfs   /dirname
     ```

     where `dirname` is the name of the file system you just changed.
- To change an exported NFS file system by using a text editor:
  1. To unexport the file system, **Enter**:

     ```
     exportfs –u   /dirname
     ```

     where `dirname` is the name of the file system you want to change.
  2. Open the **/etc/exports** file with your favorite text editor.
  3. Make whatever changes you want.
  4. Save and close the **/etc/exports** file.
  5. Re–export the file system by entering:

     ```
     exportfs   /dirname
     ```

     where `dirname` is the name of the file system you just changed.

Enabling Root User Access to an Exported File System

When a file system is exported, by default, the root user is not granted root access to that exported file systems. When a root user on one host requests access to a particular file from NFS, the user ID of the requester is mapped by NFS to the user ID of user `nobody` (`nobody` is one of the user names placed in the **/etc/password** file by default). The access rights of user `nobody` are the same as those given to the public (*others*) for a particular file. For example, if *others* only has run permission for a file, then user `nobody` can only run the file.

To enable root user access to an exported file system, follow the instructions in Changing an Exported File System. If you use the Web-based System Manager or SMIT method, specify in the HOSTS allowed root access field the name of the host to which you want to grant root access. If you edit the file with a text editor, add the qualifier `–root=hostname` to the file system entry. For example,

```
/usr/tps –root=hermes
```

specifies that the root user on host `hermes` may access the `/usr/tps` directory with root privileges.

Mounting an NFS File System Explicitly

To mount an NFS directory explicitly, use the Web–based System Manager, **wsm**, or use the following procedure:

1. Verify that the NFS server has exported the directory:

   ```
   showmount –e ServerName
   ```

   where `ServerName` is the name of the NFS server. This command displays the names of the directories currently exported from the NFS server. If the directory you want to mount is not listed, export the directory from the server.

2. Establish the local mount point using the **mkdir** command. A null (empty) directory that acts as the mount point (or place holder) of an NFS mount must be present for NFS to complete a mount successfully. This mount point can be created like any other directory, and no special attributes are needed.

3. Enter:

   ```
   mount ServerName:/remote/directory /local/directory
   ```

   where `ServerName` is the name of the NFS server, `/remote/directory` is the directory on the NFS server you want to mount, and `/local/directory` is the mount point on the NFS client.

4. On the client machine, enter the following SMIT fast path:

   ```
   smit mknfsmnt
   ```

5. Make changes to the following fields that are appropriate for your network configuration. Your configuration might not require completing all of the entries on this screen.

   Note:: If the SMIT interface is being used, press the Tab key to change to the correct value for each field, but do *not* press Enter until completing step 7.

   – PATHNAME of mount point.

   – PATHNAME of remote directory.

   – HOST where remote directory resides.

   – MOUNT now, add entry to /etc/filesystems or both?

   – /etc/filesystems entry will mount the directory on system RESTART.

   – MODE for this NFS file system.

6. Change or use the default values for the remaining entries, depending on your NFS configuration.

7. When you finish making all the changes on this screen, SMIT mounts the NFS file system.

8. When the **Command:** field shows the OK status, exit SMIT.

The NFS file system is now ready to use.

Using AutoFS to Automatically Mount a File System

AutoFS relies on the use of the **automount** command to propagate the automatic mount configuration information to the AutoFS kernel extension and start the **automountd** daemon. Through this configuration propagation, the extension automatically and transparently mounts file systems whenever a file or a directory within that file system is opened. The extension informs the **autmountd** daemon of mount and unmount requests, and the **autmountd** daemon actually performs the requested service.

Because the name–to–location binding is dynamic within the **autmountd** daemon, updates to a Network Information Service (NIS) map used by the **autmountd** daemon are transparent to the user. Also, there is no need to premount shared file systems for applications that have hard–coded references to files and directories, nor is there a need to maintain records of which hosts must be mounted for particular applications.

AutoFS allows file systems to be mounted as needed. With this method of mounting directories, all file systems do not need to be mounted all of the time; only those being used are mounted.

For example, to mount an NFS directory automatically:

1. Verify that the NFS server has exported the directory by entering:

   ```
   showmount –e   ServerName
   ```

   where `ServerName` is the name of the NFS server. This command displays the names of the directories currently exported from the NFS server.

2. Create an **AutoFS** map file. **AutoFS** mounts and unmounts the directories specified in this map file. For example, suppose you want to use **AutoFS** to mount the `/usr/local/dir1` and `/usr/local/dir2` directories as needed from the `serve1` server onto the `/usr/remote/dir1` and `/usr/remote/dir2` directories, respectively. In this example, the map file name is `/tmp/mount.map`.

   ```
   dir1            –rw              serve1:/usr/local/dir1
    dir2            –rw              serve1:/usr/local/dir2
   ```

3. Ensure that the **AutoFS** kernel entension is loaded and the **automountd** daemon is running. This can be accomplished in two ways:

   a. Using **SRC**: Issue `lssrc –s automountd`. If the **automountd** subsystem is not running, issue `startsrc –s automountd`.

   b. Using the **automount** command: Issue `/usr/bin/automount –v`.

   Define the map file using the command line interface by entering:

   ```
   /usr/sbin.automount  /usr/remote  /tmp/mount.map
   ```

   where `/usr/remote` is the **AutoFS** mount point on the client. If a user runs the **cd /usr/remote/dir1** command, the **AutoFS** kernel extension intercepts access to the directory and issues a remote procedure call to the **automountd** daemon, which mounts the **/usr/remote/dir1** directory and then allows the **cd** command to complete.

   ```
   /usr/sbin/automount /usr/remote /tmp/mount.map
   ```

   where `/usr/remote` is the mount point on the NFS client. If a user runs the **cd /usr/remote/dir1** command, the **automount** daemon mounts the **/usr/remote/dir1** directory and then allows the **cd** command to complete.

4. To stop the **automount** daemon, issue the **stopsrc –s automountd** command.

   If, for some reason, the **automountd** daemon was started without the use of **SRC**, issue:

   ```
    kill        automountd_PID
   ```

   where *automountd_PID* is the process ID of the **automountd** daemon. (Running the **ps –e** command displays the process ID of the **automountd** daemon.) The **kill** command sends a SIGTERM signal to the **automountd** daemon.

Establishing Predefined NFS Mounts

You can establish predefined NFS mounts using the Web-based System Manager Network application, or you can use one of the following procedures.

Note:: Define the **bg** (background) and **intr** (interruptible) options in the **/etc/filesystems** file when establishing a predefined mount that is mounted during system startup. Mounts that are noninterruptible and running in the foreground can hang the client if the network or server is down when the client system starts up. If a client cannot access the network or server, the user must start the machine again in maintenance mode and edit the appropriate mount requests.

- To establish predefined mounts through SMIT:
  1. Enter:

     ```
     smit mknfsmnt
     ```

  2. Specify values in this screen for each mount you want to predefine. Specify a value for each required field (those marked with an asterisk (*) in the left margin). Also specify values for the other fields or accept their default values. This method creates an entry in the **/etc/filesystems** file for the desired mount and attempts the mount.
- To establish the NFS default mounts by editing the **/etc/filesystems** file:
  1. Open the **/etc/filesystems** file with a text editor.
  2. Add entries for each of the remote file systems to be mounted when the system is started. For example:

     ```
     /home/jdoe:
      dev = /home/jdoe
      mount = false
      vfs = nfs
      nodename=mach2
      options = ro,soft
      type = nfs_mount
     ```

     This stanza directs the system to mount the **/home/jdoe** remote directory over the local mount point of the same name. The file system is mounted as read–only (`ro`). Because it is also mounted as `soft`, an error is returned in the event the server does not respond. By specifying the *type* parameter as `nfs_mount`, the system attempts to mount the **/home/jdoe** file (along with any other file systems that are specified in the `type = nfs_mount` group) when the **mount –t nfs_mount** command is issued.

     The example stanza below directs the system to mount the **/usr/games** file system at system startup time. If the mount fails, the system continues to attempt to mount in the background.

     ```
     /usr/games:
      dev = /usr/games
      mount = true
      vfs = nfs
      nodename=gameserver
      options = ro,soft,bg
      type = nfs_mount
     ```

     The following parameters are required for stanzas pertaining to NFS mounts:

`dev=` filesystem_name
Specifies the path name of the remote file system being mounted.

`mount=[` true | false `]`
If true the NFS file system is mounted when the system boots. If false , the NFS file system is not be mounted when the system boots.

`nodename=` hostname
Specifies the host machine on which the remote file system resides.

`vfs=` nfs
Specifies that the virtual file system being mounted is an NFS file system.

The following parameters are optional for stanzas pertaining to NFS mounts:

`type=` type_name	Defines the file system being mounted as part of the type_name mount group. This parameter is used with the **mount –t** command, which mounts groups of specified file systems at the same time.

`options=` options	Specifies one or more of the following options parameters:
	`biods= N`
	Specifies the number of **biod** daemons to start. The default is 6. N is an integer.
	`bg`
	Specifies to try the mount again in the background if the first mount attempt fails.

	`fg`
	Specifies to try the mount again in the foreground if the first mount attempt fails.
	`noacl`
	Disables, for this mount only, the Access Control List (ACL) support provided by the NFS journaled file system.
	When used between two systems, NFS supports access control lists. If the `noacl` option is used when mounting a file system, NFS does not use ACLs. The effect of the `noacl` option equals what happens when an NFS client on a system mounts from an NFS server that does not support ACLs.
	For more information about ACLs, refer to NFS Access Control Lists (ACL) Support on page 6-3.

	`retry= n`
	Sets the number of times to try the mount.
	`rsize= n`
	Sets the read buffer size to the number of bytes specified by n.
	`wsize= n`
	Sets the write buffer size to the number of bytes specified by n .

	`timeo= n`
	Sets the NFS time out to the tenths of a second specified by n. Use this variable to avoid situations that can occur in networks where the server load can cause inadequate response time.
	`retrans= n`
	Sets the number of NFS retransmissions to the number specified by n .
	`port= n`
	Sets the server port to the number specified by n .
	`soft`
	Returns an error if the server does not respond.

	`hard`
	Continues to try the request until the server responds.
	Note::
	When you specify a `hard` mount, it is possible that the process can hang while waiting for a response. To be able to interrupt the process and end it from the keyboard, use the `intr` variable in the mount variables.
	`intr`
	Allows keyboard interrupts on hard mounts.
	`ro`
	Sets the read–only variable.

	`rw`
	Sets the read–write variable. Use the `hard` variable along with this variable to avoid error conditions that can conflict with applications if a `soft` mount is attempted as read–write. See NFS Problem Determination on page 6-26 for information on hard– and soft–mounted problems.
	`secure`
	Specifies to use a more secure protocol for NFS transactions.
	`actimeo= n`
	Extends flush time by n seconds for both regular files and directories.
	Note::
	The attribute cache retains file attributes on the client. Attributes for a file are assigned a time to be erased. If the file is modified before the flush time, then the flush time is extended by the time since the previous modification (under the assumption that recently changed files are likely to change again soon). There are minimum and maximum flush time extensions for regular files and for directories.

	`acregmin= n`
	Holds cached attributes for at least n seconds after file modification.
	`acregmax= n`
	Holds cached attributes for no more than n seconds after file modification.
	`acdirmin= n`
	Holds cached attributes for at least n seconds after directory update.
	`acdirmax= n`
	Holds cached attributes for no more than n seconds after directory update.

	Note::
	If you do not set the following options, the kernel automatically sets them to these default values:
	`biods=6`
	`fg`
	`retry=10000`
	`rsize=8192`
	`wsize=8192`
	`timeo=7`
	`retrans=5`
	`port=NFS_PORT`
	`hard`
	`secure=off`
	`acregmin=3`
	`acregmax=60`
	`acdirmin=30`
	`acdirmax=60`

3. Remove any directory entries that you do not want to mount automatically at system startup.
4. Save and close the file.
5. Run the **mount –a** command to mount all the directories specified in the **/etc/filesystems** file.

Unmounting an Explicitly or Automatically Mounted File System

To unmount an explicitly or automatically mounted NFS directory, enter:

```
umount /directory/to/unmount
```

Removing Predefined NFS Mounts

You can remove a predefined NFS mount using the Web-based System Manager Network application, or you can use one of the following procedures.

- To remove a predefined NFS mount through SMIT:
  1. Enter:

     ```
     smit rmnfsmnt
     ```

- To remove a predefined NFS mount by editing the **/etc/filesystems** file:
  1. Enter the command: `umount /directory/to/unmount`.
  2. Open the **/etc/filesystems** file with your favorite editor.
  3. Find the entry for the directory you just unmounted, and then delete it.
  4. Save and close the file.

PC–NFS

PC–NFS is a program for personal computers that enables the personal computer to mount file systems exported by a Network File System (NFS) server. The personal computer can also request network addresses and host names from the NFS server. Additionally, if the NFS server is running the **rpc.pcnfsd** daemon, the personal computer can access authentication and print–spooling services.

You might want to configure the **rpc.pcnfsd** daemon on the following:

- Systems that perform user authentication services
- Systems that offer print–spooling
- All Network Information Service (NIS) master and slave servers.

Note:: Because NIS networks are typically configured so that PC–NFS can pick any NIS server as the default server, it is important that all servers have the **rpc.pcnfsd** daemon running. If running this daemon on all NIS servers is not practical, or if you want to limit requests to a specific server, add a **net pcnfsd** command to the **autoexec.bat** file on each personal computer to force it to use a specific NIS server. For more information, see *AIX 5L Version 5.2* Network Information Service *(NIS and NIS+) Guide*.

PC–NFS Authentication Service

By default, PC–NFS presents itself to NFS servers as the `nobody` user. With `nobody` privileges, all personal computer user files appear as owned by `nobody`, and consequently you cannot distinguish between different personal computer users. The authentication capability of the **rpc.pcnfsd** daemon allow you to monitor system resources and security by recognizing individual users and assigning them different privileges.

With the **rpc.pcnfsd** daemon running, a PC–NFS user can issue the **net name** command from a personal computer to log in to PC–NFS in the same manner as a user can log in to this operating system. The user name and password are verified by the **rpc.pcnfsd** daemon. This authentication procedure does not make a server more secure, but it does provide more control over access to files that are available through NFS.

PC–NFS Print–Spooling Service

The print–spooling service of the **rpc.pcnfsd** daemon enables personal computers running PC–NFS to print to printers not directly attached to the personal computer. Specifically, PC–NFS redirects files intended for personal computer printers to a file on an NFS server. This file is placed in a spool directory on the NFS server. The **rpc.pcnfsd** daemon then invokes the server printing facility. (The spooling directory must be in an exported file system so that PC–NFS clients can mount it.) When PC–NFS requests that the **rpc.pcnfsd** daemon print the file, it provides the following information:

- Name of the file to be printed
- Login ID of the user on the client
- Name of the printer to be used.

Configuring the rpc.pcnfsd Daemon

To configure the **rpc.pcnfsd** daemon:

1. Install PC–NFS program on your personal computer.
2. Select a location for the spool directory on the NFS server. The default spool directory is **/var/tmp**. The spool directory must have at least 100K bytes of free space.
3. Export the spool directory. Do not put access restrictions on the exported directory that could cause access problems in your network. For details of this procedure, see Exporting an NFS File System.
4. Start the **rpc.pcnfsd** daemon by following the instructions in Starting the rpc.pcnfsd Daemon.
5. Verify that the **rpc.pcnfsd** daemon is accessible by following the instructions in Verifying the rpc.pcnfsd Daemon Is Accessible.

Note:: Because printer–redirection requests sometimes cause file listings of zero length to be left in the PC–NFS spool directories, periodically clear spooling directories of these entries.

Starting the rpc.pcnfsd Daemon

To start the **rpc.pcnfsd** daemon using the default spooling directory:

1. With a text editor, uncomment the following entry in the **/etc/inetd.conf** file:

   ```
   pcnfsd sunrpc_udp udp wait root /usr/sbin/rpc.pcnfsd pcnfsd 150001 1
   ```

2. Save the file and exit the text editor.

To start the **rpc.pcnfsd** daemon using a directory that is different from the default:

1. Use a text editor to add the following entry to the **/etc/rc.nfs** file:

   ```
   if [ –f /usr/sbin/rpc.pcnfsd ] ; then
   /usr/sbin/rpc.pcnfsd –s   spooldir   ; echo ’ rpc.pcnfsd\c’
   fi
   ```

   where `spooldir` specifies the full path name of the spool directory.

2. Save the file and exit the text editor.
3. Using a text editor, comment the following entry in the **/etc/inetd.conf** file:

```
#pcnfsd sunrpc_udp udp wait root /usr/sbin/rpc.pcnfsd pcnfsd 150001 1
```

   Placing a pound sign (#) at the beginning of the line prevents the **inetd** daemon from starting the **rpc.pcnfsd** daemon using the default spool directory.

4. Start the **rpc.pcnfsd** daemon print spooler by typing the following at the command line:

   ```
   /usr/sbin/rpc.pcnfsd –s   spooldir
   ```

   where `spooldir` specifies the full path name of the spool directory.

For more information on updating the **inetd** configuration database, see Configuring the inetd Daemon on page 4-177.

Note:: The default directory that the **rpc.pcnfsd** daemon uses cannot be changed from the **inetd.conf** file.

Verifying the rpc.pcnfsd Daemon Is Accessible

To verify that the **rpc.pcnfsd** daemon is accessible, enter:

```
rpcinfo –u   host   150001
```

where `host` specifies the host name of the system on which you are configuring **rpc.pcnfsd**, and 15001 is the RPC program number of the **rpc.pcnfsd** daemon. After you enter the command, you will receive a message that the program is ready and waiting.

WebNFS

The operating system provides NFS server capability for WebNFS. Defined by Sun Microsystems, WebNFS is a simple extension of the NFS protocol that allows easier access to servers and clients through Internet firewalls.

A WebNFS–enhanced web browser can use an NFS universal resource locator (URL) to access data directly from the server. An example NFS URL is:

`nfs://www.` *`YourCompany`* `.com/`

WebNFS works in tandem with existing web–based protocols to provide data to clients.

WebNFS also takes advantage of the scalability of NFS servers.

Network Lock Manager

The network lock manager is a facility that works in cooperation with the Network File System (NFS) to provide a System V style of advisory file and record locking over the network. The network lock manager (**rpc.lockd**) and the network status monitor (**rpc.statd**) are network–service daemons. The **rpc.statd** daemon is a user level process while the **rpc.lockd** daemon is implemented as a set of kernel threads (similar to the NFS server). Both daemons are essential to the ability of the kernel to provide fundamental network services.

Note:: Mandatory or enforced locks are not supported over NFS.

Network Lock Manager Architecture

The network lock manager contains both server and client functions. The client functions are responsible for processing requests from the applications and sending requests to the network lock manager at the server. The server functions are responsible for accepting lock requests from clients and generating the appropriate locking calls at the server. The server will then respond to the locking request of the client.

In contrast to NFS, which is stateless, the network lock manager has an implicit state. In other words, the network lock manager must remember whether the client currently has a lock. The network status monitor, **rpc.statd**, implements a simple protocol that allows the network lock manager to monitor the status of other machines on the network. By having accurate status information, the network lock manager can maintain a consistent state within the stateless NFS environment.

Network File Locking Process

When an application wants to obtain a lock on a local file, it sends its request to the kernel using the **lockf**, **fcntl**, or **flock** subroutines. The kernel then processes the lock request. However, if an application on an NFS client makes a lock request for a remote file, the Network Lock Manager client generates a Remote Procedure Call (RPC) to the server to handle the request.

When the client receives an initial remote lock request, it registers interest in the server with the client’s **rpc.statd** daemon. The same is true for the network lock manager at the server. On the initial request from a client, it registers interest in the client with the local network status monitor.

Crash Recovery Process

The **rpc.statd** daemon on each machine notifies the **rpc.statd** daemon on every other machine of its activities. When the **rpc.statd** daemon receives notice that another machine crashed or recovered, it notifies its **rpc.lockd** daemon.

If a server crashes, clients with locked files must be able to recover their locks. If a client crashes, its servers must hold the client locks while it recovers. Additionally, to preserve the overall transparency of NFS, the crash recovery must occur without requiring the intervention of the applications themselves.

The crash recovery procedure is simple. If the failure of a client is detected, the server releases the failed client locks on the assumption that the client application will request locks again as needed. If the crash and recovery of a server is detected, the client lock manager retransmits all lock requests previously granted by the server. This retransmitted information is used by the server to reconstruct its locking state during a grace period. (The grace period, 45 seconds by default, is a time period within which a server allows clients to reclaim their locks.)

The **rpc.statd** daemon uses the host names kept in **/etc/sm** and **/etc/sm.bak** to keep track of which hosts must be informed when the machine needs to recover operations.

Starting the Network Lock Manager

By default, the **/etc/rc.nfs** script starts the **rpc.lockd** and **rpc.statd** daemons along with the other NFS daemons. If NFS is already running, you can verify that the **rpc.lockd** and **rpc.statd** daemons are running by following the instructions in Get the Current Status of the NFS Daemons on page 6-10. The status of these two daemons should be *active*. If the **rpc.lockd** and **rpc.statd** daemons are not active, and therefore not running, do the following:

1. Using your favorite text editor, open the **/etc/rc.nfs** file.
2. Search for the following lines:

   ```
   if [ –x /usr/sbin/rpc.statd ]; then
           startsrc –s rpc.statd
   fi
   if [ –x /usr/sbin/rpc.lockd ]; then
           startsrc –s rpc.lockd
   fi
   ```

3. If there is a pound sign (#) at the beginning of any of these lines, delete the character, then save and exit the file. Then start the **rpc.statd** and **rpc.lockd** daemons by following the instructions in Start the NFS Daemons on page 6-10.

   Note:: Sequence is important. Always start the **statd** daemon first.

4. If NFS is running and the entries in the **/etc/rc.nfs** file are correct, stop and restart the **rpc.statd** and **rpc.lockd** daemons by following the instructions in Stop the NFS Daemons on page 6-10 and Start the NFS Daemons on page 6-10.

   Note:: Sequence is important. Always start the **statd** daemon first.

If the **rpc.statd** and **rpc.lockd** daemons are still not running, see Troubleshooting the Network Lock Manager on page 6-24.

Troubleshooting the Network Lock Manager

If you receive a message on a client similar to:

```
clnttcp_create: RPC: Remote System error – Connection refused
 rpc.statd:cannot talk to statd at {server}
```

then the machine thinks there is another machine which needs to be informed that it might have to take recovery measures. When a machine restarts, or when the **rpc.lockd** and the **rpc.statd** daemons are stopped and restarted, machine names are moved from **/etc/sm** to **/etc/sm.bak** and the **rpc.statd** daemon tries to inform each machine corresponding to each entry in **/etc/sm.bak** that recovery procedures are needed.

If the **rpc.statd** daemon can reach the machine, then its entry in **/etc/sm.bak** is removed. If the **rpc.statd** daemon cannot reach the machine, it will keep trying at regular intervals. Each time the machine fails to respond, the timeout generates the above message. In the interest of locking integrity, the daemon will continue to try; however, this can have an adverse effect on locking performance. The handling is different, depending on whether the target machine is just unresponsive or semi–permanently taken out of production. To eliminate the message:

1. Verify that the **statd** and **lockd** daemons on the server are running by following the instructions in Get the Current Status of the NFS Daemons. (The status of these two daemons should be *active*.)
2. If these daemons are not running, start the **rpc.statd** and **rpc.lockd** daemons on the server by following the instructions in Start the NFS Daemons.

   Note:: Sequence is important. Always start the **statd** daemon first.

   After you have restarted the daemons, remember that there is a grace period. During this time, the **lockd** daemons allow reclaim requests to come from other clients that previously held locks with the server, so you might not get a new lock immediately after starting the daemons.

Alternatively, eliminate the message by:

1. Stop the **rpc.statd** and **rpc.lockd** daemons on the client by following the instructions in Stop the NFS Daemons.
2. On the client, remove the target machine entry from **/etc/sm.bak** file by entering:

   ```
   rm /etc/sm.bak/    TargetMachineName
   ```

   This action keeps the target machine from being aware that it might need to participate in locking recovery. It should only be used when it can be determined that the machine does not have any applications running that are participating in network locking with the affected machine.
3. Start the **rpc.statd** and **rpc.lockd** daemons on the client by following the instructions in Start the NFS Daemons.

If you are unable to obtain a lock from a client, do the following:

1. Use the **ping** command to verify that the client and server can reach and recognize each other. If the machines are both running and the network is intact, check the host names listed in the **/etc/hosts** file for each machine. Host names must exactly match between server and client for machine recognition. If a name server is being used for host name resolution, make sure the host information is exactly the same as that in the **/etc/hosts** file.
2. Verify that the **rpc.lockd** and **rpc.statd** daemons are running on both the client and the server by following the instructions in Get the Current Status of the NFS Daemons. The status of these two daemons should be *active*.
3. If they are not active, start the **rpc.statd** and **rpc.lockd** daemons by following the instructions in Start the NFS Daemons.
4. If they are active, you might need to reset them on both clients and servers. To do this, stop all the applications that are requesting locks.
5. Next, stop the **rpc.statd** and **rpc.lockd** daemons on both the client and the server by following the instructions in Stop the NFS Daemons.
6. Now, restart the **rpc.statd** and **rpc.lockd** daemons, first on the server and then on the client, by following the instructions in Start the NFS Daemons.

   Note:: Sequence is important. Always start the **statd** daemon first.

If the procedure does not alleviate the locking problem, run the **lockd** daemon in debug mode, by doing the following:

1. Stop the **rpc.statd** and **rpc.lockd** daemons on both the client and the server by following the instructions in Stop the NFS Daemons.
2. Start the **rpc.statd** daemon on the client and server by following the instructions in Start the NFS Daemons.
3. Start the **rpc.lockd** daemon on the client and server by typing:

   ```
   /usr/sbin/rpc.lockd –d1
   ```

   When invoked with the **–d1** flag, the **lockd** daemon provides diagnostic messages to syslog. At first, there will be a number of messages dealing with the grace period; wait for them to time out. After the grace period has timed out on both the server and any clients, run the application that is having lock problems and verify that a lock request is transmitted from client to server and server to client.

NFS Problem Determination

As with other network services, problems can occur on machines that use the Network File System (NFS). Troubleshooting for these problems involves understanding the strategies for tracking NFS problems, recognizing NFS–related error messages, and selecting the appropriate solutions. When tracking down an NFS problem, isolate each of the three main points of failure to determine which is not working: the server, the client, or the network itself.

Note:: See Troubleshooting the Network Lock Manager on page 6-24 for file lock problems.

Identifying Hard–Mounted and Soft–Mounted File Problems

When the network or server has problems, programs that access hard–mounted remote files fail differently from those that access soft–mounted remote files.

If a server fails to respond to a hard–mount request, NFS prints the message:

```
NFS  server  hostname  not  responding,  still  trying
```

Hard–mounted remote file systems cause programs to hang until the server responds because the client retries the mount request until it succeeds. Use the **–bg** flag with the **mount** command when performing a hard mount so if the server does not respond, the client will retry the mount in the background.

If a server fails to respond to a soft–mount request, NFS prints the message:

```
Connection  timed  out
```

Soft–mounted remote file systems return an error after trying unsuccessfully for a while. Unfortunately, many programs do not check return conditions on file system operations, so you do not see this error message when accessing soft–mounted files. However, this NFS error message prints on the console.

Identifying NFS Problems Checklist

If a client is having NFS trouble, do the following:

1. Verify that the network connections are good.
2. Verify that the **inetd**, **portmap**, and **biod** daemons are running on the client, by following the instructions in Get the Current Status of the NFS Daemons on page 6-10.
3. Verify that a valid mount point exists for the file system being mounted. For more information, see Configuring an NFS Client on page 6-11.
4. Verify that the server is up and running by running the following command at the shell prompt of the client:

   ```
   /usr/bin/rpcinfo  –p   server_name
   ```

   If the server is up, a list of programs, versions, protocols, and port numbers is printed, similar to the following:

```
program    vers  proto      port
100000        2      tcp       111    portmapper
100000        2      udp       111    portmapper
100005        1      udp      1025    mountd
100001        1      udp      1030    rstatd
100001        2      udp      1030    rstatd
100001        3      udp      1030    rstatd
100002        1      udp      1036    rusersd
100002        2      udp      1036    rusersd
100008        1      udp      1040    walld
100012        1      udp      1043    sprayd
100005        1      tcp       694    mountd
100003        2      udp      2049    nfs
100024        1      udp       713    status
100024        1      tcp       715    status
100021        1      tcp       716    nlockmgr
100021        1      udp       718    nlockmgr
100021        3      tcp       721    nlockmgr
100021        3      udp       723    nlockmgr
100020        1      udp       726    llockmgr
100020        1      tcp       728    llockmgr
100021        2      tcp       731    nlockmgr
```

If a similar response is not returned, log in to the server at the server console and check the status of the **inetd** daemon by following the instructions in Get the Current Status of the NFS Daemons on page 6-10.

5. Verify that the **mountd**, **portmap** and **nfsd** daemons are running on the NFS server by entering the following commands at the client shell prompt:

```
/usr/bin/rpcinfo  –u   server_name    mount
/usr/bin/rpcinfo  –u   server_name    portmap
/usr/bin/rpcinfo  –u   server_name    nfs
```

If the daemons are running at the server, the following responses are returned:

```
program  100005  version  1  ready  and  waiting
program  100000  version  2  ready  and  waiting
program  100003  version  2  ready  and  waiting
```

The program numbers correspond to the commands, respectively, as shown in the previous example. If a similar response is not returned, log in to the server at the server console and check the status of the daemons by following the instructions in Get the Current Status of the NFS Daemons on page 6-10.

6. Verify that the **/etc/exports** file on the server lists the name of the file system that the client wants to mount and that the file system is exported. Do this by entering the command:

```
showmount –e   server_name
```

This command lists all the file systems currently exported by the `server_name`.

Asynchronous Write Errors

When an application program writes data to a file in an NFS–mounted file system, the write operation is scheduled for asynchronous processing by the **biod** daemon. If an error occurs at the NFS server at the same time that the data is actually written to disk, the error is returned to the NFS client and the **biod** daemon saves the error internally in NFS data structures. The stored error is subsequently returned to the application program the next time it calls either the **fsync** or **close** functions. As a consequence of such errors, the application is not notified of the write error until the program closes the file. A typical example of this event is when a file system on the server is full, causing writes attempted by a client to fail.

NFS Error Messages

The following sections explain error codes that can be generated while using NFS.

nfs_server Error Message

Insufficient transmit buffers on your network can cause the following error message:

```
nfs_server:  bad  sendreply
```

To increase transmit buffers, use the Web–based System Manager, **wsm**, or the System Management Interface Tool (SMIT) fast path, **smit commodev**. Then select your adapter type, and increase the number of transmit buffers.

mount Error Messages

A remote mounting process can fail in several ways. The error messages associated with mounting failures are as follows:

`mount: ... already mounted`

The file system that you are trying to mount is already mounted.

`mount: ... not found in /etc/filesystems`

The specified file system or directory name cannot be matched.

If you issue the **mount** command with either a directory or file system name but not both, the command looks in the **/etc/filesystems** file for an entry whose file system or directory field matches the argument. If the **mount** command finds an entry such as the following:

```
/dancer.src:
            dev=/usr/src
            nodename=d61server
            type  =  nfs
            mount  =  false
```

then it performs the mount as if you had entered the following at the command line:

```
/usr/sbin/mount –n dancer –o rw,hard /usr/src /dancer.src
```

`... not in hosts database`

On a network without Network Information Service, this message indicates that the host specified in the **mount** command is not in the **/etc/hosts** file. On a network running NIS, the message indicates that NIS could not find the host name in the **/etc/hosts** database or that the NIS **ypbind** daemon on your machine has died. If the **/etc/resolv.conf** file exists so that the name server is being used for host name resolution, there might be a problem in the **named** database. See Name Resolution on an NFS Server on page 6-32 .

Check the spelling and the syntax in your **mount** command. If the command is correct, your network does not run NIS, and you only get this message for this host name, check the entry in the **/etc/hosts** file.

If your network is running NIS, make sure that the **ypbind** daemon is running by entering the following at the command line:

```
ps  –ef
```

You should see the **ypbind** daemon in the list. Try using the **rlogin** command to log in remotely to another machine, or use the **rcp** command to remote–copy something to another machine. If this also fails, your **ypbind** daemon is probably stopped or hung.

If you only get this message for this host name, check the **/etc/hosts** entry on the NIS server.

`mount: ... server not responding: port mapper failure – RPC timed out`

Either the server you are trying to mount from is down or its port mapper is stopped or hung. Try restarting the server to activate the **inetd**, **portmap**, and **ypbind** daemons.

If you cannot log in to the server remotely with the **rlogin** command but the server is up, check the network connection by trying to log in remotely to some other machine. Also check the server network connection.

`mount: ... server not responding: program not registered`

This means that the **mount** command got through to the port mapper, but the **rpc.mountd** NFS mount daemon was not registered.

`mount: access denied ...`

Your machine name is not in the export list for the file system you are trying to mount from the server.

You can get a list of the server exported file systems by running the following command at the command line:

```
showmount  –e   hostname
```

If the file system you want is not in the list, or your machine name or netgroup name is not in the user list for the file system, log in to the server and check the **/etc/exports** file for the correct file system entry. A file system name that appears in the **/etc/exports** file, but not in the output from the **showmount** command, indicates a failure in the **mountd** daemon. Either the daemon could not parse that line in the file, it could not find the directory, or the directory name was not a locally mounted directory. If the **/etc/exports** file looks correct and your network runs NIS, check the **ypbind** daemon on the server. It may be stopped or hung. For more information, see *AIX 5L Version 5.2* Network Information Service *(NIS and NIS+) Guide*.

`mount: ...: Permission denied`

This message is a generic indication that some part of authentication failed on the server. It could be that, in the previous example, you are not in the export list, the server could not recognize your machine **ypbind** daemon, or that the server does not accept the identity you provided.

Check the **/etc/exports** file on the server and, if applicable, the **ypbind** daemon. In this case, you can just change your host name with the **hostname** command and retry the **mount** command.

`mount: ...: Not a directory`

Either the remote path or the local path is not a directory. Check the spelling in your command and try to run on both directories.

`mount: ...: You are not allowed`

You must have root authority or be a member of the system group to run the **mount** command on your machine because it affects the file system for all users on that machine. NFS mounts and unmounts are only allowed for root users and members of the system group.

Identifying the Cause of Slow Access Times for NFS

If access to remote files seems unusually slow, ensure that access time is not being inhibited by a runaway daemon, a bad **tty** line, or a similar problem.

Checking Processes

On the server, enter the following at the command line:

```
ps  –ef
```

If the server appears to be operating correctly and other users are getting timely responses, make sure your **biod** daemons are running. Try the following steps:

1. Run the **ps –ef** command and look for the **biod** daemons in the display.

   If they are not running, continue with steps 2 and 3.

2. Stop the **biod** daemons that are in use by issuing the following command:

   ```
   stopsrc –x biod –c
   ```

3. Start the **biod** daemons by issuing the following command:

   ```
   startsrc –s biod
   ```

To determine if one or more **biod** daemons are not responding, the user can run **nfsstat –c** several times during the period when they suspect that one or more **biod** daemons are hung. If there is no noticeable change in the number of Remote Procedure Call (RPC) client reads or writes, one or more of the **biod** daemons are not performing their task. You can determine only that one or more **biod** daemons are inactive; you cannot determine which one is inactive.

Checking Network Connections

If the **biod** daemons are working, check the network connections. The **nfsstat** command determines whether you are dropping packets. Use the **nfsstat –c** and **nfsstat –s** commands to determine if the client or server is retransmitting large blocks. Retransmissions are always a possibility due to lost packets or busy servers. A retransmission rate of five percent or more is considered high.

The probability of retransmissions can be reduced by changing communication adapter transmit queue parameters. The System Management Interface Tool (SMIT) can be used to change these parameters.

The following values are recommended for NFS servers.

Adapter	MTU	Transmit queue
Token Ring		
4Mb	1500	50
16Mb	3900 1500 8500	40 (Increase if the **nfsstat** command times out.) 40 (Increase if the **nfsstat** command times out.) 40 (Increase if the **nfsstat** command times out.)
Ethernet	1500	40 (Increase if the **nfsstat** command times out.)

The larger MTU sizes for each token–ring speed reduce processor use and significantly improve read/write operations. Notes:

1. Apply these values to NFS clients if retransmissions persist.
2. All nodes on a network must use the same MTU size.

Setting MTU Sizes

To set MTU size, use the Web–based System Manager, **wsm**, or the SMIT fast path, **smit chif**. Select the appropriate adapter and enter an MTU value in the Maximum IP Packet Size field.

The **ifconfig** command can be used to set MTU size (and *must* be used to set MTU size at 8500). The format for the **ifconfig** command is:

```
ifconfig tr   n       NodeName   up mtu   MTUSize
```

where tr *n* is your adapter name, for example, `tr0`.

Another method of setting MTU sizes combines the **ifconfig** command with SMIT.

1. Add the **ifconfig** command for token rings, as illustrated in the previous example, to the **/etc/rc.bsdnet** file.
2. Enter the **smit setbootup_option** fast path. Toggle the Use BSD Style field to **yes**.

Setting Transmit Queue Sizes

Communication adapter transmit queue sizes are set with SMIT. Enter the **smit chgtok** fast path, select the appropriate adapter, and enter a queue size in the Transmit field.

Fixing Hung Programs

If programs hang during file–related work, the NFS server could have stopped. In this case, the following error message may be displayed:

```
NFS  server   hostname    not  responding,  still  trying
```

The NFS server (`hostname`) is down. This indicates a problem with the NFS server, the network connection, or the NIS server.

Check the servers from which you have mounted file systems if your machine hangs completely. If one or more of them is down, do not be concerned. When the server comes back up, your programs continue automatically. No files are destroyed.

If a soft–mounted server dies, other work is not affected. Programs that time out while trying to access soft–mounted remote files fail with the `errno` message, but you are still able to access your other file systems.

If all servers are running, determine whether others who are using the same servers are having trouble. More than one machine having service problems indicates a problem with the **nfsd** daemons on the server. In this case, log in to the server and run the **ps** command to see if the **nfsd** daemon is running and accumulating CPU time. If not, you might be able to stop and then restart the **nfsd** daemon. If this does not work, you have to restart the server.

Check your network connection and the connection of the server if other systems seem to be up and running.

Permissions and Authentication Schemes

Sometimes, after mounts have been successfully established, there are problems in reading, writing, or creating remote files or directories. Such difficulties are usually due to permissions or authentication problems. Permission and authentication problems can vary in cause depending on whether NIS is being used and secure mounts are specified.

The simplest case occurs when nonsecure mounts are specified and NIS is not used. In this case, user IDs (UIDs) and group IDs (GIDs) are mapped solely through the server **/etc/passwd** file and client **/etc/group** file. In this scheme, for a user named `B` to be identified both on the client and on the server as `B`, the user `B` must have the same UID number in the **/etc/passwd** file . The following is an example of how this might cause problems:

```
User  B  is  uid  200  on  client  foo.
User  B  is  uid  250  on  server  bar.
User  G  is  uid  200  on  server  bar.
```

The /home/bar directory is mounted from server bar onto client foo. If user B is editing files on the /home/bar remote file system on client foo, confusion results when he saves files.

The server bar thinks the files belong to user G, because G is UID 200 on bar. If B logs on directly to bar by using the **rlogin** command, he may not be able to access the files he just created while working on the remotely mounted file system. G, however, is able to do so because the machines arbitrate permissions by UID, not by name.

The only permanent solution to this is to reassign consistent UIDs on the two machines. For example, give B UID 200 on server bar or 250 on client foo. The files owned by B would then need to have the **chown** command run against them to make them match the new ID on the appropriate machine.

Because of the problems with maintaining consistent UID and GID mappings on all machines in a network, NIS or NIS+ is often used to perform the appropriate mappings so that this type of problem is avoided. See *AIX 5L Version 5.2* Network Information Service *(NIS and NIS+) Guide* for more information.

Name Resolution on an NFS Server

When an NFS server services a mount request, it looks up the name of the client making the request. The server takes the client Internet Protocol (IP) address and looks up the corresponding host name that matches that address. Once the host name has been found, the server looks at the exports list for the requested directory and checks the existence of the client name in the access list for the directory. If an entry exists for the client and the entry matches exactly what was returned for the name resolution, then that part of the mount authentication passes.

If the server is not able to perform the IP address–to–host–name resolution, the server denies the mount request. The server must be able to find some match for the client IP address making the mount request. If the directory is exported with the access being to all clients, the server still must be able to do the reverse name lookup to allow the mount request.

The server also must be able to look up the correct name for the client. For example, if there exists an entry in the **/etc/exports** file like the following:

```
/tmp      –access=silly:funny
```

the following corresponding entries exist in the **/etc/hosts** file:

```
150.102.23.21        silly.domain.name.com
150.102.23.52        funny.domain.name.com
```

Notice that the names do not correspond exactly. When the server looks up the IP address–to–host–name matches for the hosts silly and funny, the string names do not match exactly with the entries in the access list of the export. This type of name resolution problem usually occurs when using the **named** daemon for name resolution. Most **named** daemon databases have aliases for the full domain names of hosts so that users do not have to enter full names when referring to hosts. Even though these host–name–to–IP address entries exist for the aliases, the reverse lookup might not exist. The database for reverse name lookup (IP address to host name) usually has entries containing the IP address and the full domain name (not the alias) of that host. Sometimes the export entries are created with the shorter alias name, causing problems when clients try to mount.

Limitations on the Number of Groups in the NFS Structure

On systems that use NFS Version 3.2, users cannot be a member of more than 16 groups without complications. (Groups are defined by the **groups** command.) If a user is a member of 17 or more groups, and the user tries to access files owned by the 17th (or greater) group, the system does not allow the file to be read or copied. To permit the user access to the files, rearrange the group order.

Mounting from NFS Servers That Have Earlier Version of NFS

When mounting a file system from a pre–Version 3 NFS server onto a Version 3 NFS client, a problem occurs when the user on the client executing the mount is a member of more than eight groups. Some servers are not able to deal correctly with this situation and deny the request for the mount. The solution is to change the group membership of the user to a number less than eight and then retry the mount. The following error message is characteristic of this group problem:

```
RPC:    Authentication  error;  why=Invalid  client  credential
```

Problems That Occur If the NFS Kernel Extension Is Not Loaded

Some NFS commands do not execute correctly if the NFS kernel extension is not loaded. Some commands with this dependency are: **nfsstat**, **exportfs**, **mountd**, **nfsd**, and **biod**. When NFS is installed on the system, the kernel extension is placed in the **/usr/lib/drivers/nfs.ext** file. This file is then loaded as the NFS kernel extension when the system is configured. The script that does this kernel extension loads the **/etc/rc.net** file. There are many other things done in this script, one of which is to load the NFS kernel extension. It is important to note that Transmission Control Protocol/Internet Protocol (TCP/IP) kernel extension should be loaded before the NFS kernel extension is loaded.

Note:: The **gfsinstall** command is used to load the NFS kernel extension into the kernel when the system initially starts. This command can be run more than once per system startup and it will not cause a problem. The system is currently shipped with the **gfsinstall** command used in both the **/etc/rc.net** and **/etc/rc.nfs** files. There is no need to remove either of these calls.

NFS Reference

List of Network File System (NFS) Files

bootparams	Lists clients that diskless clients can use for booting.
exports	Lists the directories that can be exported to NFS clients.
networks	Contains information about networks on the Internet network.
pcnfsd.conf	Provides configuration options for the **rpc.pcnfsd** daemon.
rpc	Contains database information for Remote Procedure Call (RPC) programs.
xtab	Lists directories that are currently exported.
/etc/filesystems	Lists all the file systems that are mounted at system restart.

List of NFS Commands

chnfs	Starts a specified number of **biod** and **nfsd** daemons.
mknfs	Configures the system to run NFS and starts NFS daemons.
nfso	Configures NFS network options.
automount	Mounts an NFS file system automatically.
chnfsexp	Changes the attributes of an NFS–exported directory.
chnfsmnt	Changes the attributes of an NFS–mounted directory.
exportfs	Exports and unexports directories to NFS clients.
lsnfsexp	Displays the characteristics of directories that are exported with NFS.
lsnfsmnt	Displays the characteristics of mounted NFS systems.
mknfsexp	Exports a directory using NFS.
mknfsmnt	Mounts a directory using NFS.
rmnfs	Stops the NFS daemons.
rmnfsexp	Removes NFS–exported directories from a server’s list of exports.
rmnfsmnt	Removes NFS–mounted file systems from a client’s list of mounts.

List of NFS Daemons

Locking Daemons

lockd	Processes lock requests through the RPC package.
statd	Provides crash–and–recovery functions for the locking services on NFS.

Network Service Daemons and Utilities

biod	Sends the client read and write requests to the server.
mountd	Answers requests from clients for file system mounts.
nfsd	Starts the daemons that handle a client requests for file system operations.
pcnfsd	Handles service requests from PC–NFS clients.
nfsstat	Displays information about the ability to receive calls for a particular machine.
on	Executes commands on remote machines.
portmap	Maps RPC program numbers to Internet port numbers.
rexd	Accepts request to run programs from remote machines.
rpcgen	Generates C code to implement an RPC protocol.
rpcinfo	Reports the status of RPC servers.
rstatd	Returns performance statistics obtained from the kernel.
rup	Shows the status of a remote host on the local network.
rusers	Reports a list of users logged on to the remote machines.
rusersd	Responds to queries from the **rusers** command.
rwall	Sends messages to all users on the network.
rwalld	Handles requests from the **rwall** command.
showmount	Displays a list of all clients that have mounted remote file systems.
spray	Sends a specified number of packets to a host.
sprayd	Receives packets sent by the **spray** command.

Secure Networking Daemons and Utilities

chkey	Changes the user encryption key.
keyenvoy	Provides an intermediary between user processes and the key server.
keylogin	Decrypts and stores the user secret key.
keyserv	Stores public and private keys.
mkkeyserv	Starts the **keyserv** daemon and uncomments the appropriate entries in the **/etc/rc.nfs** file.
newkey	Creates a new key in the **publickey** file.
rmkeyserv	Stops the **keyserv** daemon and comments the entry for the **keyserv** daemon in the **/etc/rc.nfs** file.
ypupdated	Updates information in Network Information Service (NIS) maps.

For additional information on NFS security, see Network File System (NFS) Security in *AIX 5L Version 5.2 Security Guide*.

Sun Diskless Client Support

bootparamd	Provides information necessary for booting to diskless clients.

NFS Subroutines

cbc_crypt, **des_setparity**, or **ecb_crypt**	Implements Data Encryption Standard (DES) routines.

SMBFS

Server Message Block Filesystem (SMBFS) allows access to shares on SMB servers as a local filesystems on AIX. In this filesystem, the user can create, delete, read, write, and modify the access times of files and directories. The owner or access mode of files and directories cannot be changed.

SMBFS can be used to access files on an SMB server. The SMB server is a server running Samba; an AIX server running AIX Fast Connect; or a Windows XP, Windows NT, or Windows 2000 server or workstation. Each of these server types allows a directory to be exported as a share. This share can then be mounted on an AIX system using SMBFS.

Install SMBFS

To install SMBFS on an AIX system, install the **bos.cifs_fs** package.

When the **bos.cifs_fs** package in installed, the device `nsmb0` is created. This device allows the **mount** command to establish a connection between the SMB server and the client.

Mount the Filesystem

The directory can be mounted in one of two ways. It can be performed through the AIX **mount** command. For example:

```
mount –v cifs –n pezman/user1/pass1 –o uid=201,fmode=750 /home /mnt
```

For more information on the **mount** command and for explanations of the flags used, see the **mount** command in *AIX 5L Version 5.2 Commands Reference*.

You can also mount the filesystem by using the SMIT utility, `smit cifs_fs`, which will run the **mount** command after gathering all necessary information.

In order to mount an SMBFS filesystem, it is necessary to provide a user name and password to authenticate to the server. This user name and password will be used to perform all necessary file operations on the server. The **Password** field in the smit panel is not marked as required. However, if that field is not filled out, the user will be prompted for a password, through the standard AIX password prompt. This way, the user can provide a password without making it viewable.

Whenever a filesystem command, such as read, is invoked on a file inside the SMBFS mount point, a request is sent to the server to read the file. The user name and password are sent as part of this request so that the server can determine whether the user has permission on the server to perform a read operation on that file. Therefore, ultimate authority lies with the server as to whether an operation on a file is permissible.

However, the `fmode` option provides a way for the root user on the client system to control access to the files on the server before the server is queried. If the `fmode` option is not provided by the user, the default is `755`. The following table illustrates how the `fmode` option works using a write request:

Case number	**user authenticated to server**	**user on client side wanting write access**	**mount owner, group, and mode**	**owner, group, and mode on server**	**access allowed**
Case 1	user1	user2	user1, staff rwxr–xr–x	user1, staff rwxrwxr–x	no
Case 2	user1	root	user1, staff rwxr–xr–x	user2, staff rwxr–xr–x	no

Case 3	user1	user1	user1, staff rwxr–xr–x	user2, staff rwxrwxr–x	yes
Case 4	user1	user1	user, staff rwxr–xr–x	root, system rwx———	no
Case 5	user1	user1	user1, staff rwxr–xr–x	root, system rwxrwxrwx	yes

In Case 1, access was denied because owner, group, and mode at mount on client did not allow write access to `user2`.

In Case 2, access was denied because, even though `root` has access to everything on the client side, the server–authenticated user, `user1`, does not have access to the file on the server.

In Case 3, access was granted because `user1` was the owner at mount, and `user1`, being a member of group `staff` on the server, had access to the file on the server.

In Case 4, access was denied because, even though `user1` was the owner at mount, the file is owned by `root` on the server, with no access by group or other.

In Case 5, access was granted because `user1` was the owner at mount, and `user1` had access to the file on the server through other permissions.

Troubleshooting SMBFS

If the **mount** command or **smit cifs_fs** fastpath returns an error, consider the following:

- Make sure the user name and password are correct. The user name and password need to allow access to the share on the server.
- Make sure that the server name is correct. If the server name is correct, use the fully–qualified hostname in case the server is not part of the same subnet as the client. You may also try using the server IP address
- Make sure that `lsdev –L|grep nsmb` command returns a device name. If an `nsmb` device is not available, then the AIX client will not be able to establish a connection to the SMB server.
- Make sure that the share name is correct. If the share does not exist on the server or cannot be accessed with the user name and password given, the SMB serer will reject the connection request.
- Use event id 525 to collect system trace data for SMBFS.

Chapter 7. TTY Devices and Serial Communications

This chapter contains information on managing tty terminal devices. Topics discussed are:

TTY Overview

A tty terminal device is a character device that performs input and output on a character–by–character basis. The communication between terminal devices and the programs that read and write to them is controlled by the tty interface. Examples of tty devices are:

- Modems
- ASCII terminals
- System console (LFT)
- **aixterm** under AIXwindows

The tty devices can be added, deleted, listed, and changed like any other device on your system by using the Web-based System Manager Devices application, the SMIT tool, or device–specific commands.

TERM Values for Different Displays and Terminals

Information about terminal capabilities is stored in the **terminfo** database. The value of the **TERM** environment variable identifies the specific terminal description in the **terminfo** database. This provides all information that a program needs for communicating effectively with the current tty device.

TERM values for various terminals

Display/Terminal	**Value**
3161 ASCII Terminal	ibm3161
3163 ASCII Terminal	ibm3161
DEC VT100 (terminal)	vt100
DECVT220	vt220
3151 ASCII Display Station with Cartridge or 3161 ASCII Display Station with Cartridge	ibm3161–C
3162 ASCII Display Station	ibm3161
3162 ASCII Display Station with Cartridge	ibm3162
6091 Display	lft
AIXwindows	aixterm

For information about the entries in the **terminfo** database, see the **terminfo** file format. To convert **termcap** entries into **terminfo** entries, see the **captoinfo** command. (The **termcap** file contains the terminal descriptions for older Berkeley systems.)

Setting TTY Characteristics

The *line discipline* provides the hardware–independent user interface for communicating between the computer and an asynchronous device. For example, a user is able to erase a single line or to interrupt a currently running process by typing a particular sequence of characters. You can define the meaning of these character sequences as well as set other terminal characteristics, such as the communication speed, by using the Web-based System Manager Devices application, the **chdev** command, the System Management Interface Tool (SMIT), or the **stty** command.

Setting Attributes on the Attached TTY Device

For correct communication between the host and an attached tty device, the following are required:

- A properly wired communications cable
- Matching communications values (line speed, character size, parity, stop bit, and interface) between the host and the attached tty device.

Managing TTY Devices

Managing TTY devices tasks

Task	**SMIT fast path**	**Command or file**	Web-based System Manager Management Environment
List Defined TTY Devices	**smit lsdtty**	**lsdev –C –c tty –H**	Software —> **Devices** —> **All Devices**.
Add a TTY	**smit mktty**	**mkdev –t tty** [1,2]	Software —> **Devices** —> **Overview and Tasks**.
Move a TTY to Another Port [3]	**smit movtty**	**chdev –l** *Name* **–p** *ParentName* **–w** *ConnectionLocation* [2,4]	
Change/Show Characteristics of a TTY	**smit chtty**	**lsattr –l** *Name* **–E** (to show); **chdev –l** *Name* (to change) [4,5]	Software —> **Devices** —> **All Devices** —> **Selected** —> **Properties**.
Remove a TTY [3]	**smit rmtty**	**rmdev –l** *Name*	Software —> **Devices** —> **All Devices** —> **Selected** —> **Delete**.
Configure a Defined TTY (Make Available for Use)	**smit mktty**	**mkdev –l** *Name*	Software —> **Devices** —> **Overview and Tasks**.

Notes:

1. Other flags may be used to further specify the new tty device. For example, to define and configure an RS–232 tty device connected to port `0` on the 8–port asynchronous adapter `sa3` with the `speed` attribute set to `19200` and other attributes set to values retrieved from the `foo` file:

   ```
   mkdev –t tty –s rs232 –p sa3 –w 0 –a speed=19200 –f foo
   ```

2. The **mkdev** and **chdev** commands support options that are not possible with Web-based System Manager or SMIT.
3. Disable the tty before doing this task. See the **pdisable** command.
4. Use flags to change specific characteristics about a tty from the command line.
5. You can select a Posix baud rate from the List function, or you can type in non–Posix baud rate directly into the field. If the selected baud rate cannot be supported by the modem’s hardware, the system displays an error message.

If adding or changing a tty from the command line, consult the following list to find out the *Attribute* name to specify in the **–a** *Attribute* **=** *Value* flag for the characteristic you want to set. For example, specify `–a speed=Value` to set the baud rate of a tty device.

Characteristic	Attribute Name
Enable LOGIN	login
BAUD rate speed	speed
PARITY	parity
BITS per character	bpc
Number of STOP BITS	stops
TIME before advancing to next port setting	timeout
XON–XOFF handshaking	xon
TERMINAL type	term
FLOW CONTROL to be used	flow_disp
OPEN DISCIPLINE to be used	open_disp
STTY attributes for RUN time	runmodes
STTY attributes for LOGIN	logmodes
RUN shell activity manager	shell
LOGGER name	logger
STATUS of device at BOOT time	autoconfig
TRANSMIT buffer count	tbc
RECEIVE trigger level	rtrig
STREAMS modules to be pushed at open time	modules
INPUT map file	imap
OUTPUT map file	omap
CODESET map file	csmap
INTERRUPT character	intr
QUIT character	quit
ERASE character	erase
KILL character	kill
END OF FILE character	eof
END OF LINE character	eol
2nd END OF LINE character	eol2
DELAY SUSPEND PROCESS character	dsusp
SUSPEND PROCESS character	susp
LITERAL NEXT character	lnext
START character	start
STOP character	stop
WORD ERASE character	werase
REPRINT LINE character	reprint
DISCARD character	discard

Dynamic Screen Utility

The dynamic screen utility, or **dscreen** command, is a utility that allows a single physical terminal to be connected to several virtual terminal sessions (screens) at one time. It is mainly intended for use with terminals that have two or more pages of screen memory (for example, the 3151 Models 310 or 410 display with the Cartridge for Expansion). With such terminals, switching between virtual screens also switches between physical terminal screen pages allowing each virtual screen's image to be saved and restored. On terminals without multiple pages of screen memory, the **dscreen** command can still be used to switch among virtual screen sessions although the appearance of the screen will not be maintained.

Note:: For full support of **dscreen** utility, the terminal must be able to switch internal screen pages on command and must remember the cursor position for each page. While the **dscreen** utility will work on both smart and dumb terminals, screen images are not saved during screen changes on dumb terminals.

dscreen Terminal Configuration Information File

The **dscreen** utility terminal configuration information file (or **dsinfo** file) is used to define a different set of keys to be used with the **dscreen** utility. This might be done, for example, when the originally defined **dscreen** utility keys conflict with a software application in use on the system.

The **dsinfo** file terminal type assumes a single page of screen memory. Therefore, if a terminal supports additional pages of screen memory, the **dsinfo** file must be customized to use the appropriate sequence for page memory control. Consult the appropriate terminal reference guide for the specific control sequence.

The default **dsinfo** file is **/usr/lbin/tty/dsinfo**. Use the **–i** flag to specify a different **dsinfo** file. This remainder of this section refers to the default file. However, the same information applies to any customized **dsinfo** file you create.

For more information concerning the **dsinfo** file, see Dynamic Screen Assignment.

Key Action Assignments

When the **dscreen** command is executed, it starts a virtual screen. Some of the keys on the terminal keyboard are not passed through to the virtual screen; instead, the **dscreen** utility intercepts these keys and performs certain actions when they are pressed. The actions include:

Select (see Select Keys on page 7-7)	Selects a specified screen.
Block (see Block Keys on page 7-7)	Blocks all input and output.
New (see New Keys on page 7-7)	Starts a new screen session.
End (see End and Quit Keys on page 7-7)	Ends the **dscreen** utility.
Quit (see End and Quit Keys on page 7-7)	Quits the **dscreen** utility.
Previous (see Previous Key on page 7-7)	Switches to previous screen.
List (see List Key on page 7-7)	Lists the **dscreen** assigned keys and their actions.

The function of each key is dependent on the terminal and the terminal description in the **/usr/lbin/tty/dsinfo** file.

Select Keys

When a new virtual screen is created, it is assigned a select key. Pressing the select key causes the following actions:

- A switch from the physical terminal to the video page associated with the particular virtual screen.
- Input and output is directed appropriately between the physical terminal and the virtual screen.

After all of the select keys defined in the **dsinfo** file have virtual screens assigned to them, no more screens can be created. Individual screen sessions end when the original shell process exits. This frees the associated select key for use with another virtual screen. The **dscreen** utility is ended when there are no more active screens.

Block Keys

Block keys are used to stop output in a fashion similar to Ctrl–S key when using IXON flow control. The purpose of these keys is to allow for transparently setting up terminal sessions on two computers using a terminal that has two serial ports.

New Keys

Pressing a new screen key creates a new logical screen and assigns it to one of the select keys. Each new screen requires:

- A select key as defined in the dsinfo file
- A **dscreen** pseudo–terminal device
- Enough memory for the various structures used in screen tracking
- A process to run the shell from.

If any of these are not available, the new screen operation fails with a message indicating the reason for the failure.

End and Quit Keys

Pressing an end key causes the following to occur:

- Send a **SIGHUP** signal to all the screen sessions
- Clean up
- Exit with a status of 0.

Pressing a quit key performs the same actions but exits with status of 1.

Previous Key

Pressing a previous key switchs the terminal to the screen that was last displayed. Notes:

1. Do not switch screens when the current screen is being written to; an escape sequence might be truncated and leave the terminal in an unknown state.
2. Some terminal displays can save the cursor position for individual screens but might not save other states such as insert mode, inverse video, and so forth. If this is the case, users should avoid these modes while switching screens.

List Key

Pressing a list key displays a list of keys and their actions on the terminal display. Only those keys recognized by the **dscreen** utility will be shown. When a new screen is created using the **dscreen** utility, the message `Press KEY for help`, where `KEY` is the name of the list key displayed on the terminal. Note that the message is displayed *only* if there is a list key defined.

Dynamic Screen Assignment

The terminal description entry in the **/usr/lbin/tty/dsinfo** file has the same number of screen selection keys as the terminal has physical screen pages. If more screen selection keys are defined than the number of physical screen pages, the **dscreen** utility will dynamically assign physical screen pages to virtual screens.

When a virtual screen is selected that does not have an associated page of screen memory, the **dscreen** utility assigns the least recently used physical screen to the virtual screen. Depending on the specifications maintained in the **/usr/lbin/tty/dsinfo** description file, an indication that the physical screen is connected to a different virtual screen might be noticeable; for example, the screen is cleared.

dsinfo File

The **dsinfo** file is a database of terminal descriptions used by the **dscreen** multiple screen utility. The file contains the following information:

- The **dscreen** utility keys and the functions they perform
- Number of screen memory pages for the terminal
- Code sequences sent or received to use the above features.

The terminal type entries in the default **dsinfo** file resemble the following 3151 ASCII terminal values:

```
# The Cartridge for Expansion (pn: 64F9314) needed for this entry
 ibm3151|3151|IBM 3151,
 dsks=\E!a^M|Shift–F1|,           # Selects first screen
 dsks=\E!b^M|Shift–F2|,           # Selects second screen
 dsks=\E!c^M|Shift–F3|,           # Selects third screen
 dsks=\E!d^M|Shift–F4|,           # Selects fourth screen
 dskc=\E!e^M|Shift–F5|,           # Creates a new screen
 dske=\E!f^M|Shift–F6|\E pA\EH\EJ, # Go to screen 1 and end
 dskl=\E!g^M|Shift–F7|,           # Lists function keys (help)
 dskp=\E!h^M|Shift–F8|,           # Go to previous screen
 dskq=\E!i^M|Shift–F9|\E pA\EH\EJ, # Go to screen 1 and quit
 dsp=\E pA|\EH\EJ,                # Terminal sequence for screen 1
 dsp=\E pB|\EH\EJ,                # Terminal sequence for screen 2
 dsp=\E pC|\EH\EJ,                # Terminal sequence for screen 3
 dsp=\E pD|\EH\EJ,                # Terminal sequence for screen 4
 dst=10,                          # Allow 1 second timeout buffer
```

Entry Format for dsinfo

Entries in the **dsinfo** file consist of comma–separated fields. The first field is a list of alternate names for the terminal, each name is separated by a pipe (**|**) character. Any text preceded by a pound (#) character is regarded as a comment and ignored by **dscreen**. The remaining fields are strings describing the capabilities of the terminal to the **dscreen** utility. Within these strings, the following escape codes are recognized:

Escape Sequence	**Description**
\E,\e	escape character
\n,\l	newline (or linefeed) character
\r	carriage return
\t	tab character
\b	backspace character
\f	formfeed character
\s	space character
\ *nnn*	character with octal value *nnn*
^ *x*	Ctrl– *x* for any appropriate *x* value

Any other character preceded by a backslash will yield the character itself. The strings are entered as *type=string*, where *type* is the type of string as listed below, and *string* is the string value.

It is important that the entry fields in the **dsinfo** file be separated by commas. If a comma is omitted or truncated from the end of a **dsinfo** file entry, the file will become unreadable by the **dscreen** utility and an error will be returned to the display.

String Types

The string types are as follows:

dskx

A string type that starts with dsk describes a key. The type must be four letters long, and the fourth letter *x* indicates what action is taken when the key is received. The key types are:

Type
Action

dsks
Switch Screens

dskb
Block Input and Output

dske
End **dscreen**

dskq
Quit **dscreen** (exit status=1)

dskc
Create New Screen

dskp
Switch to Previous Screen

dskl
List Keys and Actions

Any other key type (that is, a string type dsk *x* that does not end in s, b, e, q, p, or l) will cause no internal **dscreen** action, but will show up in the key listing and will be recognized and acted on. A type of dskn (n for No Operation) should be used when no internal **dscreen** action is desired.

The value string for each key has three substrings, which are separated by pipe (**|**) characters.

Note:
Use \ **|** to include the **|** character in one of the substrings.

The first substring is the sequence of characters that the terminal sends when the key is pressed. The second substring is a label for the key that is printed when a list of keys is displayed. The third substring is a sequence of characters that **dscreen** sends to the terminal when this key is pressed before performing the action this key requests.

dsp

A string type of dsp describes a physical screen in the terminal. One dsp string should be present for each physical screen in the terminal. The value string for each physical screen has two substrings, which are separated by a pipe (|) character.

The first substring is the sequence of characters to send to the terminal to display and output to the physical page on the terminal.

The second substring is sent to the terminal when the page is used for something new. This second substring is often set to the clear screen sequence. It is sent under the following two conditions:

1. When a new virtual terminal session is being created.
2. When there are more virtual terminals than there are physical screens. If a virtual terminal is selected that requires **dscreen** to reuse one of the physical screens, it will send this sequence to the screen to indicate that the screen contents do not match the output of the virtual terminal connected.

Note:
Running with more virtual terminals than physical screens can be confusing and is not recommended; it can be avoided by defining no more screen selection keys (dsks=) than physical screens (dsp=) in the dsinfo entry.

dst A

String with a type of dst adjusts **dscreen** 's input timeout. The value of the string is a decimal number. The timeout value is in tenths of seconds and has a maximum value of 255 (default= **1** [or 0.1 seconds]).

When **dscreen** recognizes a prefix of an input key sequence but does not have all the characters of the sequence, it will wait for more characters to be sent until it is recognizable. If the timeout occurs before more characters are received, the characters are sent on to the virtual screen and **dscreen** will not consider these characters as part of an input key sequence.

It may be necessary to raise this value if one or more of the keys **dscreen** is to trigger on is actually a number of keystrokes (that is assigning Ctrl–Z 1, Ctrl–Z 2, Ctrl–Z 3, etc., for screen selection and Ctrl–Z N for new screen and so on).

Example 1

The following example **/usr/lbin/tty/dsinfo** entry is for a Wyse–60 with three screen sessions:

```
wy60|wyse60|wyse model 60,
 dsks=^A‘^M|Shift–F1|,
 dsks=^Aa^M|Shift–F2|,
 dsks=^Ab^M|Shift–F3|,
 dskc=\200|Ctrl–F1|,
 dske=\201|Ctrl–F2|\Ew0\E+,
 dskl=\202|Ctrl–F3|,
 dsp=\Ew0|\E+,
 dsp=\Ew1|\E+,
 dsp=\Ew2|\E+,
```

With this entry:

- Shift–F1 through Shift–F3 are used for selecting screens 1 through 3.
- Ctrl–F1 creates a new screen.
- Ctrl–F2 sends: `Esc w 0 Esc +` to the screen (switching to window 0 and clearing the screen) and then ends **dscreen**.
- Ctrl–F3 lists the keys and their functions.

Each time a physical screen is used for a new screen, the sequence `Esc +` will be sent to the terminal, which will clear the screen.

Example 2

This example is for a Wyse–60 with three screen sessions, but one of the screens is on a second computer communicating through the second serial port on the terminal:

```
wy60–1|wyse60–1|wyse model 60 – first serial port
 dsks=^A‘^M|Shift–F1|,
 dsks=^Aa^M|Shift–F2|,
 dsks=^Ab^M|Shift–F3|\Ed#^Ab\r^T\Ee9,
 dskc=\200|Ctrl–F1|,
 dske=\201|Ctrl–F2|\Ed#\201^T\Ew0\E+,
 dskl=\202|Ctrl–F3|,
 dsp=\Ew0|\E+,dsp=\Ew1|\E+,
 wy60–2|wyse60–2|wyse model 60 – second serial port
 dsks=^A‘^M|Shift–F1|\Ed#^A‘\r^T\Ee8,
 dsks=^Aa^M|Shift–F2|\Ed#^Aa\r^T\Ee8,
 dsks=^Ab^M|Shift–F3|,
 dskc=\200|Ctrl–F1|,
 dske=\201|Ctrl–F2|\Ed#\201^T\Ew0\E+,
 dskl=\202|Ctrl–F3|,
 dsp=\Ew2|\E+,
```

dscreen must be run on both computers, with terminal type wy60–1 on the first computer and terminal type wy60–2 on the second computer (using the **–t** option to **dscreen**). The wy60–1 entry will be examined first.

The first two key entries are unchanged from the original wy60 entry. The third key, however, has type dskb, which means block both input and output. When this key is pressed, the sequence:

```
Esc d # Ctrl–A b CR Ctrl–T Esc e 9
```

is sent to the terminal; after this output is blocked and **dscreen** continues scanning input for key sequences but discards all other input.

The sequence `Esc d #` puts the terminal in transparent print mode, which echoes all characters up to a Ctrl–T out through the other serial port.

The characters `Ctrl–A b CR` are sent out the other serial port, informing the **dscreen** process on the other computer that it should activate the window associated with the Shift–F3 key.

The Ctrl–T key sequence exits the transparent print mode. The Esc 9 key sequence causes the terminal to switch to the other AUX serial port for data communications.

At this point, the other computer takes over, sends an `Esc w 2` to switch to the third physical screen, and then resumes normal communication.

The wy60–2 entry follows the same general pattern for keys Shift–F1 and Shift–F2:

- Switch to transparent print mode
- Send function key string to other computer
- Switch transparent print off
- Switch to the other serial port

The end key, Ctrl–F2, works the same for both computers; it sends the end key sequence to the other computer through the transparent print mechanism, switches the terminal to window 0, clears the screen, then exits.

Modems

Modems provide serial communications across ordinary telephone lines. This section discusses modem standards, general modem setup, and specific configuration tips for popular modems.

Modem Overview

A *modem* is a device that allows you to connect one computer to another across ordinary telephone lines. The current telephone system is incapable of carrying the voltage changes required for a direct digital connection. A modem overcomes this limitation by modulating digital information into audio tones for transmission across the phone line, and by demodulating those tones back into digital information on reception. Modems are commonly used with Basic Network Utilities (BNU) or other implementations of the UNIX–to–UNIX Copy Program (UUCP). A high–speed (14,400 bps or greater) modem can be used with Serial Line Interface Protocol (SLIP) to provide Transmission Control Protocol/Internet Protocol (TCP/IP) connectivity as well.

Often, the term *baud* is used to refer to modem speed instead of bps. Baud is actually a measurement of the modulation rate. In older modems, only 1 bit was encoded in each signal change, so modem baud rate was equal to modem speed. Modems that operate at higher speeds, however, still generally operate at 2,400 (or even 1,200) baud, and encode two or more bits per signal change. A modem's bps rate is calculated by multiplying the number of data bits per signal with the baud (for example, 2,400 baud x 6 bits per signal change = 14,400 bits per second). Most modern modems can communicate at a variety of speeds (for example, 28,800, 14,400, 9,600, 7,800, 4,800, and 2,400 bps).

Telecommunications Standards

The older speeds of 300, 1,200, and 2,400 bps were well defined. However, as modem manufacturers began to devise methods for gaining higher speeds, each modem manufacturer started to use a proprietary method incompatible with modems from other manufacturers. Today, the ITU–TSS (formerly the United Nations Consultative Committee for International Telephony and Telegraphy, abbreviated CCITT) defines standards for most high–speed communications.

Even high–speed modems are much slower than other methods of computer communication. A high–speed modem can operate at 28,800 bps, but an Ethernet connection operates at 10,000,000 bps. To boost data throughput, high–speed modems typically offer one or more data compression algorithms. These algorithms can boost the throughput of a high–speed modem to speeds of 57,600 bps (if the data rate is 14,400 bps) or 115,200 bps (if the data rate is 28,800 bps). Note that these compression algorithms are sensitive to the data being transmitted. If the data has already been compressed (for example, with the **compress** command), the data compression methods of high–speed modems offer little or no benefit, and might even reduce data throughput. When using a modem with data compression technology, the speed of the data terminal equipment/data circuit–terminating equipment (DTE/DCE) connection between the computer and the modem is equal or greater than the nominal data rate of the connection between modems. For example, with a V.32 *bis* modem with V.42 *bis* data compression, the data rate of the modem (the speed at which the modem communicates across telephone lines) is 14,400 bps. When the V.42 *bis* compression is active, actual data throughput can reach 57,600 bps. To accommodate the greater throughput offered by data compression, the speed of the link between the computer and the modem should be set to 57,600 bps.

The ITU–TSS defines standards for high–speed communications, including data compression algorithms. ITU–TSS standards are usually named V. *nn*, where *nn* is a number. Another, slightly less common standard is the Microcom Networking Protocol (MNP). Available in versions (called classes) 1–9, MNP is a high–performance, high–speed protocol that was available relatively early, and became something of a de facto standard before the advent of the CCITT standards.

ITU–TSS Communications Standards

Following is a list of some common communications standards defined by the ITU–TSS. Note that this only a partial list. For a complete list, refer to the Internet website for the International Telecommunication Union.

V.29 ITU–TSS standard for half–duplex 9600 bps communications.

V.32 ITU–TSS standard for full–duplex 9600 bps communications.

V.32 *bis* ITU–TSS standard for 14,400 communications. V.32 *bis* is a revision to the V.32 standard.

V.34 ITU–TSS standard for 33,600 bps communications. Note that this standard achieves 33,600 bps data rates using multiple bit encoding, instead of the data compression scheme used by MNP Class 9. This standard was previously referred to as V. *fast*.

V.42 ITU–TSS error correcting procedures for DCEs using asynchronous to synchronous conversion.

V.42 bis Revised ITU–TSS data compression standard.

MNP Communications Standards

MNP Class 1 An asynchronous, half–duplex, byte–oriented method of transferring data realizing about 70% efficiency. Uncommon in modern modems.

MNP Class 2 A full–duplex counterpart to MNP Class 1. Uncommon in modern modems.

MNP Class 3 A synchronous, bit–oriented full–duplex method of transferring data realizing about 108% efficiency. (Efficiency greater than 100% is realized because the start/stop bits required for an asynchronous connection are eliminated. The link between the modem and the system are still asynchronous).

MNP Class 4 An enhancement to MNP Class 3 including a mechanism for varying the packet size (adaptive packet assembly) and a means of eliminating redundant administrative overhead (data phase optimization). An MNP Class 4 modem offers approximately 120% efficiency.

MNP Class 5 Class 5 includes data compression along with Class 4 features. An MNP Class 5 modem offers 200% efficiency.

MNP Class 6 MNP Class 6 allows incorporation of multiple, incompatible modulation techniques into one modem (universal link negotiation). This allows MNP Class 6 modems to begin communication at a slower speed and negotiate a transition to a higher speed. Class 6 also includes a statistical duplexing scheme that dynamically allocates utilization of half–duplex modulation to simulate full–duplex service. All features of MNP Class 5 are supported.

MNP Class 7 Incorporates enhanced data compression. Combined with Class 4, efficiencies of 300% can be realized.

MNP Class 8	N/A
MNP Class 9	Combines enhanced data compression with V.32 technology to allow data rates up to 28,800 bps.

Generic Modem Setup

To set up a modem:

1. Create a TTY device on the Operating System
2. Attach the Modem with Appropriate Cables
3. Add a TTY for the Modem
4. Configure the Modem

Create a TTY device on the Operating System

Use the System Management Interface Tool (SMIT) to define a tty port for the device attachment. Most fields are for the general device type. The only field that can affect the modem is the Enable LOGIN field with the following values:

DISABLE	No getty process is run on the port. Use this setting for dial–out only modem ports.
ENABLE	A getty process is run on the port. Use this setting for dial–in modems only.
SHARE	A getty process is run on the port, but the getty process allows programs to dial in and out of this port without manually changing to disable or enable. Use this setting for bidirectional port usage.
DELAY	A getty is run on the port in bi–directional mode, but no herald is sent until the getty process receives a keystroke from the user.

Fields specific to the 128–port asynchronous adapter:

Force Carrier or Ignore Carrier Detect	disable*
Perform Cooked Processing in Adapter	disable

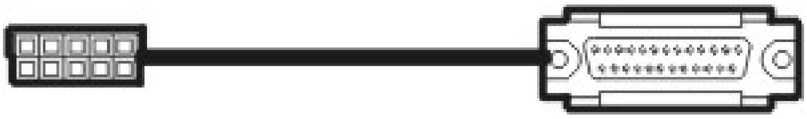
Note:: This setting indicated by an asterisk (*) is set to disabled if the 10 pin RJ–45 connector is used. This setting should be enabled if the 8 pin RJ–45 connector is used.

Attach the Modem with Appropriate Cables

The first step in setting up a modem is to attach the modem with the appropriate cables. Part numbers and their descriptions are listed below.

6323741	Async Cable, EIA–232; used to attach all asynchronous devices; sometimes used with other cable assemblies.
59F3740	10 to 25–pin D–shell connector used to attach asynchronous cable 6323741 to native serial ports S1 and S2 as shown in the following figure.

Figure 29. 10 to 25–Pin Connector This illustration shows a 10 to 25–pin connector.

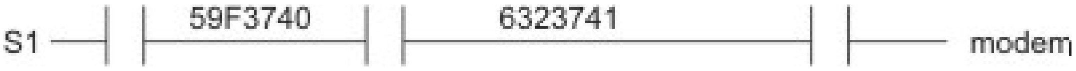


59F3432 Cable P used to connect to 16–port concentrator. Part number includes four RJ–45 to DB–25 converter cables.

Following are some examples of cable connections:

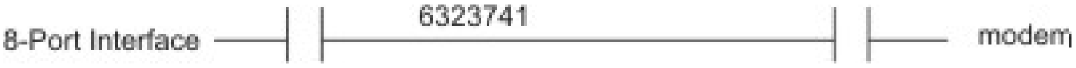
1. To attach a modem to native serial port S1, use the following cables:

   Figure 30. Modem to Native Serial Port Cable Assembly This illustration shows a 59F3740 cable on the serial port end and a 6323741 on the modem end.



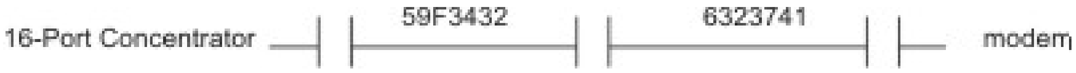
2. To attach a modem to an 8–port async adapter (EIA–232) interface cable assembly, use the following cables:

   Figure 31. 8–Port Interface to Modem Cable Assembly This illustration shows an 8–port interface connected to a modem with a 6323741 cable.



3. To attach a modem to a 16–port concentrator on a 64–port adapter, use the following cables:

   Figure 32. 16–Port Interface to Modem Cable Assembly This illustration shows a 59F3432 cable on the serial port end and a 6323741 on the modem end.



Add a TTY for the Modem

First, ensure that the system is turned on and that the modem is turned off. Use the Web–based System Manager, **wsm**, or the SMIT fast path **smit mktty**.

Configure the Modem

Use only one of the two methods presented in this section for configuring the modem. If you have Basic Networking Utilities (BNU) installed, see Sending AT Commands with the cu Command. If you do not have BNU installed, see Sending AT Commands Using a C Program. For information on installing BNU, see Basic Networking Utilities.

Sending AT Commands with the cu Command

If you have the Basic Network Utilities (BNU) installed, use the **cu** command to configure a modem as follows. The commands and settings discussed in this section configure a Hayes–compatible modem with the basic parameters needed for operation on the server’s serial ports.

1. Add the following line to your **/usr/lib/uucp/Devices** file. Do not add the line if it is already in the file. (Replace # with the number for your port.)

   ```
   Direct tty# – Any direct
   ```

2. Verify that the tty is disabled by typing the following::

   ```
   pdisable tty#
   ```

3. Type the following command:

   ```
   cu –ml tty#
   ```

   You should see a message that says Connected.

4. Verify that you have the attention of the modem by typing the following:

   ```
   AT
   ```

   The modem should respond with OK. If it didn’t, refer to Troubleshooting Modem Problems.

   For additional **AT** commands and their descriptions, see AT Command Summary.

5. Depending on which getty option you selected, enter one of the following commands. Substitute the tty device for *n*.

   - `penable tty` *n*
   - `pshare tty` *n*
   - `pdelay tty` *n*
   - `pdisplay tty` *n*

   The modem is now configured with the basic commands needed to perform most of the operating system’s serial communications needs. If you have problems, invoke the **cu –dl** to start a diagnostic trace on the connection.

Sending AT Commands Using a C Program

If the previous method failed, or if you do not have BNU installed, try running the following C program. Create a file called **motalk.c** containing the following code. Save the file. Compile and run it according to the instructions in the program comments.

```
/*************************************************************/
/*  MoTalk – A ”C” program for modem setup.                  */
/*           This program is meant as an aid only and is     */
/*           not supported by IBM.                           */
/*                  compile:  cc –o motalk motalk.c          */
/*                  Usage:  motalk /dev/tty? [speed]         */
/*************************************************************/
 #include <errno.h>
 #include <stdio.h>
 #include <signal.h>
 #include <fcntl.h>
 #include <termio.h>
 FILE *fdr, *fdw;
 int fd;
 struct termio term_save, stdin_save;
 void Exit(int sig)
 {
    if (fdr) fclose(fdr);
    if (fdw) fclose(fdw);
    ioctl(fd, TCSETA, &term_save);
    close(fd);
    ioctl(fileno(stdin), TCSETA, &stdin_save);
    exit(sig);
 }
 main(int argc, char *argv[])
 {
    char *b, buffer[80];
    int baud=0, num;
    struct termio term, tstdin;
    if (argc < 2 || !strcmp(argv[1], ”–?”))
    {
       fprintf(stderr, ”Usage: motalk /dev/tty? [speed]\n”);
       exit(1);
    }
    if ((fd = open(argv[1], O_RDWR | O_NDELAY)) < 0)
    {
       perror(argv[1]);
       exit(errno);
    }
    if (argc > 2)
    {
       switch(atoi(argv[2]))
```

```
       {
          case   300: baud = B300;
                      break;
          case  1200: baud = B1200;
                      break;
          case  2400: baud = B2400;
                      break;
          case  4800: baud = B4800;
                      break;
          case  9600: baud = B9600;
                      break;
          case 19200: baud = B19200;
                      break;
          case 38400: baud = B38400;
                      break;
          default:    baud = 0;
                      fprintf(stderr, ”%s: %s is an unsupported baud\n”,
argv[0],argv[2]);
                      exit(1);
          }
       }
    /* Save stdin and tty state and trap some signals */
    ioctl(fd, TCGETA, &term_save);
    ioctl(fileno(stdin), TCGETA, &stdin_save);
    signal(SIGHUP, Exit);
    signal(SIGINT, Exit);
    signal(SIGQUIT, Exit);
    signal(SIGTERM, Exit);
    /*  Set stdin to raw mode, no echo */
    ioctl(fileno(stdin), TCGETA, &tstdin);
    tstdin.c_iflag = 0;
    tstdin.c_lflag &= ~(ICANON | ECHO);
    tstdin.c_cc[VMIN] = 0;
    tstdin.c_cc[VTIME] = 0;
    ioctl(fileno(stdin), TCSETA, &tstdin);
    /*  Set tty state */
    ioctl(fd, TCGETA, &term);
    term.c_cflag |= CLOCAL|HUPCL;
    if (baud > 0)
    {
       term.c_cflag &= ~CBAUD;
       term.c_cflag |= baud;
    }
    term.c_lflag &= ~(ICANON | ECHO); /* to force raw mode */
    term.c_iflag &= ~ICRNL; /* to avoid non–needed blank lines */
    term.c_cc[VMIN] = 0;
    term.c_cc[VTIME] = 10;
    ioctl(fd, TCSETA, &term);
    fcntl(fd, F_SETFL, fcntl(fd, F_GETFL, 0) & ~O_NDELAY);
    /*  Open tty for read and write */
    if ((fdr = fopen(argv[1], ”r”)) == NULL )
    {
       perror(argv[1]);
       exit(errno);
    }
    if ((fdw = fopen(argv[1], ”w”)) == NULL )
    {
       perror(argv[1]);
       exit(errno);
    }
    /*  Talk to the modem */
    puts(”Ready... ^C to exit”);
    while (1)
    {
       if ((num = read(fileno(stdin), buffer, 80)) > 0)
          write(fileno(fdw), buffer, num);
       if ((num = read(fileno(fdr), buffer, 80)) > 0)
          write(fileno(stdout), buffer, num);
       Exit (0);
    }
 }
```

Hayes and Hayes–Compatible Modems

1. Change the tty settings, if necessary, using the Web–based System Manager, **wsm**, or the SMIT fast path, **smit chtty**. For example, you might want to change the Enable LOGIN field to **Share** or **Enable**.
2. Add the following line to **/usr/lib/uucp/Systems** file:

   ```
   hayes Nvr HAYESPROG 2400
   ```
3. Add this to **/usr/lib/uucp/Devices** file:

   ```
   # For programming the  hayes modem only:
   HAYESPROG tty0 – 2400 HayesProgrm2400
   #regular ACU entry:
   ACU tty0 – Any hayes
   ```
4. Add this to **/usr/lib/uucp/Dialers** file:

   ```
   # This Entry is used to PROGRAM the modem ONLY:
   # the next 3 lines should be made into one:
   HayesProgrm2400     =,–,     ”” \d\dAT\r\c OK AT&F\r\c OK ATM1\r\c OK
   AT&D3\r\c OK AT&K3&C1\r\c OK ATL0E0Q2\r\c OK ATS0=1\r\c OK AT&W\r\c
   OK
   hayes     =,–,     ”” \dAT\r\c OK ATDT\T\d\r\c CONNECT
   ```
5. To program the modem, enter the command `cu –d hayes`. This command uses the **cu** command to program the modem. Because no connection is made to another system, the command will fail. The modem is programmed if `sendthem AT&W` and then `OK got it` appear in the output.

   If you are not doing binary file transfers or using BNU, leave out the **&K3** command, and set XON as the flow control to be used. However, it is more efficient to use hardware flow control (as opposed to XON–XOFF handshaking). To do that, use the settings and the `Dialers` entries from the next step.
6. After the modem is programmed, you can configure the system device driver to use hardware flow control. Using Web–based System Manager, **wsm** or SMIT (**smit chtty** fast path), change the flow control to RTS. Check your modem manuals to find out whether your modem supports hardware flow control.

Troubleshooting Modem Problems

This section attempts to identify common problems when using a modem with your computer. Keep in mind the following points:

- Some modems are case–sensitive. Use uppercase letters for the **AT** commands.
- In normal operation, it is preferable for the modem to reset when the DTR is dropped (&D3 setting). When the modem is being set up for the first time, however, it is advisable not to have the modem reset if the DTR is dropped (&D2 setting). If the modem resets itself, all programmed settings that are not saved in the modem’s memory will be lost.

  Not having the modem reset also protects changes when &C1 is set. Changing the Carrier Detect status may cause the carrier detect line to toggle on some modems, which causes the **cu** command to drop the line. You may want to set up the modem to &D3 after the final setup is made.
- Although the commands given in this manual are standard for most Hayes–compatible modems, there is no guarantee that they are standard for your modem. Compare the commands with your modem manual before proceeding.

Symptom	Cause	Solution
The modem (or other device attached to the serial port) causes the system to gradually slow down and eventually hang. Turning off the device usually lets the system function normally again.	An intelligent modem has CD always ON. The system senses this and sends a login herald, which the modem tries to interpret as a command. The modem fails to recognize the login herald as a valid command, and echoes back to the tty port on the system. This cycle repeats continuously.	Set the tty port to delay on the system so no login herald will be sent. With this setting, only a valid carriage return character from the host logging in will cause a login herald to be sent. You can also change the modem's AT set profile to set CD to ON only when a valid carrier is sensed on the telephone line.

Software Services Modem Questionnaire

Before calling for assistance with modem problems, please collect the following information:

- Level of the operating system. How long have you been at this level of the operating system?
- Has the modem ever worked before?
- What type of modem are you using? What type of modem is on the other end of the telephone connection?
- To what adapter type (64–port, 128–port, S1,...) is the modem attached?
- To which port number is the modem attached?
- To which tty number is the modem attached?
- What type of cabling are you using?
- What is the login setting (share, delay, enable)?
- Can the modem connect to other modems?
- Can other modems connect to your modem?
- What are the following values in Web–based System Manager, SMIT, modem, or port?
  - XON/XOFF?
  - RTS/CTS?
  - BPS rate?
- Include the following in your problem description:
  - Does the port lock intermittently?
  - Can you dial out? Can others dial in?
  - Any other specific and descriptive error conditions.
- Are there errors on the console? What are they?
- Are there errors in the error report? (**errpt** or **errpt –a**)
- What command are you using to dial out?
- What software is involved on the system?

AT Command Summary

The following is a summary of the Hayes Smartmodem command set. These commands comprise the AT command set used by many popular modems. This information comes from the Hayes Smartmodem 2400 *Quick Reference Card*, published by Hayes Microcomputer Products, Inc. Consult the modem documentation for a list of relevant AT commands.

AT	Command prefix – precedes command line.
<CR>	Carriage return (newline) character – terminated the command line.
A	Go off–hook, remain in command mode.
A/	Repeat previous command line. This command is not preceded with **AT** or followed by **<CR>** /.
B0	Select CCITT V.22 standard for 1200 bps communications.
B1	Select Bell 212A standard for 1200 bps communications.
D	Enter originate mode, dial the number that follows, and attempt to go online. D is usually followed by T for tone, P for pulse may also be used.
DS = *n*	Dial the number stored in location *n.*
E0	Disable character echo in the command state.
E1	Enable character echo in the command state.
H0	Go on–hook (hang up the phone).
H1	Operate switch–hook and auxiliary relay.
I0	Return product identification code.
I1	Perform checksum on firmware ROM; return checksum.
I2	Perform checksum on firmware ROM; returns `OK` or `ERROR` as the result.
L0	Speaker off.
L1	Low speaker volume.
L2	Medium speaker volume.
L3	High speaker volume.
M0	Speaker off.
M1	Speaker on until carrier detected.
M2	Speaker always on.
M3	Speaker on until carrier detected, except during dialing.
O0	Enter online state.
O1	Enter online state and initiate equalizer retrain.
Q0	Modem returns result codes.
Q1	Modem does not return result codes.
Sr	Set pointer to register r.
Sr = *n*	Set register r to value *n.*
V0	Display result codes in numeric form.
V1	Display result codes in verbose form (as words).
X0	Enable features represented by result codes 0–4.
X1	Enable features represented by result codes 0–5, 10.
X2	Enable features represented by result codes 0–6, 10.

X3	Enable features represented by result codes 0–5, 7, 10.
X4	Enable features represented by result codes 0–7, 10.
Y0	Disable long space disconnect.
Y1	Enable long space disconnect.
Z	Reset modem
&C0	Assume data carrier always present.
&C1	Track presence of data carrier.
&D0	Ignore DTR signal.
&D1	Assume command state when an on–to–off transition of DTR occurs.
&D2	Hang up and assume command state when an on–to–off transition of DTR occurs.
&D3	Reset when an on–to–off transition of DTR occurs.
&F	Recall the factory settings as the active configuration.
&G0	No guard tone.
&G1	500 Hz guard tone.
&G2	1800 Hz guard tone.
&J0	RJ–11/RJ41/RJ45S telco jack.
&J1	RJ–11/RJ–13 telco jack.
&P0	Pulse dial with make/break ratio 39/61.
&P1	Pulse dial with make/break ratio 33/67.
&Q0	Operate in asynchronous mode.
&Q n	Operate in synchronous mode *n.*
&R0	Track CTS according to RTS.
&R1	Ignore RTS; always assume presence of CTS.
&S0	Assume presence of DSR signal.
&S1	Track presence of DSR signal.
&T0	Terminate test in progress.
&T1	Initiate local analog loopback.
&T3	Initiate digital loopback.
&T4	Grant request from remote modem for remote data link (RDL).
&T5	Deny request from remote modem for RDL.
&T6	Initiate remote digital loopback.
&T7	Initiate remote digital loopback with self–test.
&T8	Initiate local analog loopback with self–test.
&V	View active configuration, user profiles, and stored numbers.
&W n	Save storable parameters of active configuration as user profile *n*.
&X0	Modem provides transmit clock signal.
&X1	Data terminal provides transmit clock signal.
&X2	Receive carrier provides transmit clock signal.
&Y n	Recall user profile *n*.
&Z n= *x*	Store phone number *x* in location *n*.

S–Register Summary

Register	Range	Description
S0	0–255	Select number of rings before answer.
S1	0–255	Ring count (incremented with each ring).
S2	0–127	Define escape sequence character (ASCII).
S3	0–127	Define carriage return character (ASCII).
S4	0–127	Define line feed character (ASCII).
S5	0–32, 127	Define backspace character (ASCII).
S6	2–255	Select wait–time in seconds before blind dialing.
S7	1–55	Select wait–time in seconds for carrier/dial tone.
S8	0–255	Select duration in seconds of comma.
S9	1–255	Carrier detect response time in .1 second increments (10 = 1 second).
S10	1–255	Delay between carrier loss and hang–up in .1 second increments.
S11	50–255	Duration/spacing of tones in milliseconds.
S12	50–255	Escape sequence guard time in .02 second intervals.
S13	—	Reserved.
S14	—	Reserved.
S15	—	Reserved.
S16	—	Reserved – functions for this register are controlled by the **&T** commands).
S17	—	Reserved.
S18	0–255	Test timer duration in seconds.
S19	—	Reserved.
S20	—	Reserved.
S21	—	Reserved.
S22	—	Reserved.
S23	—	Reserved.
S24	—	Reserved.
S25	0–255	Select DTR change detect time in .01 second intervals.

S26	0–255	RTS to CTS delay in .01 second intervals.
S27	—	Reserved.

Result Codes Summary

Number	**Word**	**Description**
0	OK	Command executed.
1	CONNECT	Connection established at 0–300 bps.
2	RING	Ring signal detected.
3	NO CARRIER	Carrier signal lost or not detected.
4	ERROR	Invalid command, checksum, error in command line, or command line too long.
5	CONNECT 1200	Connection established at 1200 bps.
6	NO DIALTONE	No dial tone detected.
7	BUSY	Busy signal detected.
8	NO ANSWER	No response when dialing a system.
9	CONNECT 2400	Connection established at 2400 bps.

Dial modifiers

The following lists and describes dial modifiers:

0–9 # * A–D	Digits and characters for dialing.
P	Pulse dial.
T	Tone dial.
,	Delay processing of next character.
!	Hookflash.
@	Wait for silence.
W	Wait for dial tone.
;	Return to command state after dialing.
R	Reverse mode.
S = *n*	Dial number stored at location *n.*

Setting Terminal Options with stty–cxma

stty–cxma is a utility program that sets and displays the terminal options for the PCI 8– and PCI 128–port adapters and is located in `/usr/lbin/tty` directory. The format is:

```
stty–cxma [–a] [option(s)] [ttyname]
```

With no options, **stty–cxma** displays all special driver settings, modem signals, and all standard parameters displayed by stty(1) for the tty device referenced by standard input. Command options are provided to change flow control settings, set transparent print options, force modem control lines, and display all tty settings. Any unrecognized options are passed to stty(1) for interpretation. The options are:

–a Displays all of the unique adapter option settings, as well as all of the standard tty settings reported by the **stty –a** command.

ttyname Sets and displays options for the given tty device, instead of standard input. This form can be used with a tty pathname prefixed by `/dev/` or with a simple tty name beginning with tty. This option may be used on a modem control line when no carrier is present.

The following options specify transient actions to be performed immediately:

break Sends a 250 ms break signal out on the tty line.

flush Indicated an immediate flush (discard) of tty input and output.

flushin Flushes tty input only.

flushout Flushes tty output only.

The following options specify actions that are reset when the device is closed. The device will use the default values the next time it is opened.

stopout Stops output exactly as if an XOFF character was received.

startout Restarts stopped output exactly as if an XON character was received.

stopin Activates flow control to stop input.

startin Releases the flow control to resume stopped input.

[–]dtr [drop] Raises the DTR modem control line, unless DTR hardware flow control is selected.

[–]rts [drop] Raises the RTS modem control line, unless RTS hardware flow control is selected.

The following options remain in effect until the system is rebooted or until the options are changed.

[–]fastcook Performs cooked output processing on the intelligent card to reduce host CPU usage, and increase raw mode input performance.

[–]fastbaud Alters the baud rate tables, so 50 baud becomes 57,600 baud, 75 baud becomes 76,800 baud, 110 baud becomes 115,200 baud, and 200 baud becomes 230,000 baud for supported devices.

[–]rtspace Enables/disables RTS hardware input flow control, so RTS drops to pause remote transmission.

[–]ctspace Enables/disables CTS hardware output flow control, so local transmission pauses when CTS drops.

[–]dsrpace Enables/disables DSR hardware output flow control, so local transmission pauses when DSR drops.

[–]dcdpace Enables/disables DCD hardware output flow control, so local transmission pauses when DCD drops.

[–]dtrpace Enables/disables DTR hardware input flow control, so DTR drops to pause remote transmission.

[–]forcedcd Disable [re–enable] carrier sense, so the tty may be opened and used even when carrier is not present.

[–]altpin Maps the RJ–45 connector pinouts to the default 10–pin connector values or the 8–pin connector values. When this parameter is **enabled**, the location of DSR and DCD is switched so that DCD is available when using an 8–pin RJ–45 connector instead of the 10–pin RJ–45 connector. (Default= **disable**.)

Possible values:

enable (specifies 8–pin connector values)

disable (specifies 10–pin connector values)

startc *c* Sets the XON flow control character. The character may be given as a decimal, octal, or hexadecimal number. Octal numbers are recognized by the presence of a leading zero, and hexadecimal numbers are denoted by a leading 0x. For example, the standard XON character, CTRL–Q, can be entered as 17 (decimal), 021 (octal), or 0x11 (hexadecimal).

stopc *c* Sets the XOFF flow control character. The character may be given as a decimal, octal, or hexadecimal number (see **startc** for format of octal and hexadecimal numbers).

astartc *c* Sets auxiliary XON flow control character. The character may be given as a decimal, octal, or hexadecimal number (see **startc** for format of octal and hexadecimal numbers).

astopc *c* Sets auxiliary XOFF flow control character. The character may be given as a decimal, octal, or hexadecimal number (see **startc** for format of octal and hexadecimal numbers).

[–]aixon Enables auxiliary flow control, so that two unique characters are used for XON and XOFF. If both XOFF characters are received, transmission will not resume until both XON characters are received.

[–]2200flow Uses 2200 style flow control on the port. The 2200 terminals support an attached printer and use four flow control characters: terminal XON (0xF8), printer XON (0xF9), terminal XOFF (0xFA) and printer XOFF (0xFB).

[–]2200print Determines how these flow control characters are interpreted. If 2200print is set, run independent flow control for terminal and transparent print devices. Otherwise, terminal and printer flow control are logically tied together. If either XOFF character is received, all output is paused until the matching XON character is received.

maxcps *n* Sets the maximum characters per second (cps) rate that characters are output to the transparent print device. The rate chosen should be just below the average print speed. If the number is too low, printer speed will be reduced. If the number is too high, the printer uses flow control, and user entry time is reduced. The default is 100 cps.

maxchar *n* Sets the maximum number of transparent print characters the driver places in the output queue. Reducing this number increases system overhead; increasing this number delays operator keystroke echo times when the transparent printer is in use. The default is 50 characters.

bufsize *n* Sets the driver's estimate of the size of the transparent printer's input buffer. After a period of inactivity, the driver bursts this many characters to the transparent printer before reducing to the maxcps rate. The default is 100 characters.

onstr *s* Sets the terminal escape sequence to turn transparent printing on. The strings can be composed of standard ASCII printing and non–printing characters. Control (non–printing) characters must be entered by their octal values, and must consist of three digits preceded by a back–slash character. For example, the Escape character, 33 octal, should be entered as \033. If transparent printing is turned on by the string <Esc>[5i (ANSI standard), it would be entered as: \033[5i.

offstr *s* Sets the terminal escape sequence to turn transparent printing off. Refer to **onstr** *s* for the format of the strings.

term *t* Sets the transparent printer on/off strings to values found in the internal default table. Internal defaults are used for the following terminals: adm31, ansi, dg200, dg210, hz1500, mc5, microterm, multiterm, pcterm, tvi, vp–a2, vp–60, vt52, vt100, vt220, wyse30, wyse50, wyse60, or wyse75. If the terminal type is not found in the internal default table, ditty reads the terminfo entry for the terminal type and sets transparent print on/off strings to values given by the mc5/mc4 attributes found in the terminfo entries.

ATE Overview

The Asynchronous Terminal Emulation (ATE) program, an optional software product, enables a system to emulate a terminal on a remote system. Using ATE, you can log in to most systems that support asynchronous terminals, including any system that supports RS–232C or RS–422A connections. You can set ATE parameter values so the remote system recognizes your terminal as either an attached workstation terminal or a DEC VT100 terminal.

Setting Up ATE Overview

Before you run ATE, you must install the software and set up the ports and connections. ATE uses both direct (cabled) connections and modem connections. Local RS–232C connections allow a maximum distance of 15 meters (50 feet) between machines, and RS–422A connections allow up to 1200 meters (4000 feet) between machines.

Before you use ATE to call a remote system, be sure that the remote system tty device is ready to accept a call. If another user on the remote system uses ATE to call your terminal, be sure your tty device is ready to accept the call.

See Setting Up ATE for more information about installing and setting up ATE.

Note:: You must be a member of a UNIX–to–UNIX Copy Program (UUCP) group to use ATE. A user with root authority can use Web-based System Manager or the System Management Interface Tool (SMIT) to set up a UUCP group.

Customizing ATE

The first time you run ATE, the program creates an **ate.def** default file in the current directory. The **ate.def** file contains parameters the ATE program uses for:

- Data transmission characteristics
- Local system features
- Dialing directory file
- Control keys.

To change the defaults, edit the **ate.def** file.

If you need to run ATE with different settings, you can maintain **ate.def** files in different directories. You can then run ATE from the appropriate directory depending on the settings needed for specific sessions. However, running ATE from many directories requires multiple copies of the **ate.def** file, which uses system storage.

See How to Edit the ATE Default File in *AIX 5L Version 5.2 System User's Guide: Communications and Networks* for details about editing the **ate.def** file.

You can temporarily change settings without modifying the default file. To do this, use the **alter** and **modify** subcommands. Settings you change with the **alter** or **modify** subcommand remain in effect until you exit the program with the **quit** subcommand. When you exit ATE, the settings return to the defaults set in the **ate.def** file.

When installed, ATE uses the **/usr/lib/dir** system–wide dialing directory file. You can temporarily change settings in the dialing directory file for a specific modem connection. Settings changed in this way revert to the default when the connection ends, rather than when you exit ATE. A user with root authority can modify the **/usr/lib/dir** file to include numbers for modems used by everyone on the system. Individual users can also create their own dialing directory files and modify their copies of the **ate.def** file to make ATE use those directories.

The How to Set up an ATE Dialing Directory article in *AIX 5L Version 5.2 System User's Guide: Communications and Networks* explains how to set up ATE to use a customized dialing directory.

You can edit the dialing directory file to include frequently used phone numbers Additionally, you can change the baud rate, data character length, stop bits, parity, echoing, and line–feeds for a phone number if these characteristics differ from the defaults. If a number is not in the directory file, you can complete the connection by using the **connect** subcommand.

Note:: A dialing directory file can contain up to 20 lines (one entry per line). ATE ignores subsequent lines.

Changing ATE Characteristics

The following table identifies the ATE characteristics that the user can change and the appropriate methods for changing each characteristic.

Note:: All ATE characteristics can be changed in the **ate.def** file.

Changing ATE characteristics

Characteristic	**Change with**
Control keys	**ate.def** file
Data character length	**alter** subcommand or dialing directory entry
Dialing directory file name	**directory** subcommand
Echoing (on or off)	**modify** subcommand or dialing directory entry
File name for incoming data (capture file)	**modify** subcommand
Final dial suffix for the modem	**alter** subcommand
Initial dial prefix for the modem	**alter** subcommand
Line feeds	**modify** subcommand or dialing directory entry
Number of redialing attempts	**alter** subcommand
Number of stop bits	**alter** subcommand or dialing directory entry
Parity (even or odd)	**alter** subcommand or dialing directory entry
Port name (device)	**alter** subcommand
Rate (bits per second)	**alter** subcommand or dialing directory entry
Telephone number	dialing directory entry
Transfer protocol (pacing or xmodem)	**alter** subcommand
Type of pacing (character or interval)	**alter** subcommand
VT100 emulation (on or off)	**modify** subcommand
Wait time between redialing attempts	**alter** subcommand
Write (capture) incoming data to a file	**modify** subcommand
Xon/Xoff protocol (on or off)	**modify** subcommand

Setting Up ATE

This article provides information on setting up Asynchronous Terminal Emulation (ATE).

Prerequisites

- The ATE program must be installed on the system. ATE is an optional program product.
- The user must have root user authority to set up the port for the communications device.

Procedure

To prepare ATE to run on the system:

1. Install an asynchronous adapter card in an appropriate slot in the system unit, unless the system has a built–in serial port.
2. Plug the RS–232C or RS–422A cable into the adapter card or the built–in serial port.
3. Add a tty device for the communications port. To do this, use the Web–based System Manager, **wsm**, or enter:

   ```
   smit mktty
   ```
4. Select Add a TTY.
5. Select the tty type.
6. Select a parent adapter
7. Select a port.
8. Set the Enable LOGIN field to **disable**.
9. Set Terminal Type to **HFT** or **dumb**.
10. Make the necessary adjustments for the environment. The most common changes are line speed, parity settings, number of bits per character, and whether the line is to be driven as a remote or local line. Use `BPC 8` and `no parity` if National Language Support (NLS) is required.
11. Set up the port for the device.
    - To set up a port to call out with ATE, use the **pdisable** command. For example, to set up port `tty1`, enter:

      ```
      pdisable tty1
      ```
    - To set up a port so that others can call in, use the **penable** command. For example, to let other systems call in to the `tty2` port, enter:

      ```
      penable tty2
      ```
12. Ensure the device has previously been defined to the remote system. Once the device is defined, the ATE program must be customized to reflect the device settings on the remote system. Customize the default settings with the **alter** and **modify** subcommands or by editing the **ate.def** default file. To change the default settings for a telephone connection, use a dialing directory file entry.

TTY Troubleshooting

This section discusses troubleshooting the tty subsystem:

- Respawning Too Rapidly Errors on page 7-30
- Error Log Information and TTY Log Identifiers on page 7-31
- Clear a Hung tty Port on page 7-35

Respawning Too Rapidly Errors

The system records the number of getty processes spawned for a particular tty in a short time period. If the number of getty processes spawned in this time frame exceeds five, then the `Respawning Too Rapidly` error is displayed on the console and the port is disabled by the system.

The tty stays disabled for about 19 minutes or until the system administrator enables the port again. At the end of the 19 minutes, the system automatically enables the port, resulting in the spawning of a new **getty** process.

Possible Causes

- Incorrect modem configuration
- A port is defined and enabled but no cable or device is attached to it
- Bad cabling or loose connection
- Noise on communication line
- Corruption of, or tampering with, **/etc/environment** or **/etc/inittab** files
- tty configuration is corrupted
- Hardware is defective

Each of these possible causes is explained in Procedures for Recovery on page 7-30.

Procedures for Recovery

- Incorrect modem configuration:

  Ensure that the modem carrier detect is *not* forced high.

  Note:: The following applies to Hayes–compatible modems

  1. Connect to the modem and examine the active profile.
  2. Set the modem carrier detect to **&C1** rather than **&C0** (forced high). Use the following AT modem commands to set and change the carrier attribute:

     ```
     AT&C1
      AT&W
      Notes:
     ```

     a. See Sending AT Commands with the cu Command

     b. See your modem documentation for further information.

- Disable the tty, remove the tty definition, or attach a device to the port:

  – To disable the tty definition use the **chdev** command as follows:

  ```
  chdev –l   ttyName   –a Login=disable
  ```

  After running this command, the tty does *not* become enabled after a system restart.

  – To remove the tty definition:

  1. Disable the tty port, use the **pdisable** command, enter:

     ```
     pdisable   ttyName
     ```

2. Remove the tty definition from the system. See Managing TTY Devices for further information.

- Check for bad cables or loose connections:

  1. Check cabling. Tighten loose connections and replace damaged or inappropriate connectors.
  2. Verify that the suspected cabling is serial cable P/N 6323741 or that the cable meets the same standard. Replace damaged or inappropriate cables.

- Eliminate noise on communication line:

  1. Verify that cabling is correct length and impedance.
  2. Ensure that toroid rings are in place where needed on longer cables.
  3. Check routing of cables; they should not be close to fluorescent lights or motors.

- Check for corruption of, or tampering with, the **/etc/environment** or the **/etc/inittab** files:

  1. If possible, compare these files against known good copies.
  2. Copy the files as a backup and make changes as needed.
  3. In the **/etc/environment** file, remove any lines that are *not*:
     - blank lines
     - comment lines
     - *variable* = *value*
  4. In the **/etc/inittab** file, examine the tty devices lines. If the tty is set to off, it is likely that the tty port is not being used. If it is not being used, remove the tty definition or attach a device to the port.

- Remove corrupted tty configuration:

  1. Remove the tty definition. Use the Web-based System Manager Devices application or see Managing TTY Devices for further information.
  2. If you want a hard copy record of the tty definition before removing it, press the Image key (F8 or Esc+8). This will capture the current screen image and copy it to the **smit.log** file in your **$HOME** directory.
  3. Read the tty definition. See the instructions for Adding a TTY under Managing TTY Devices.

- Locate defective hardware:

  1. Run diagnostics using the **diag** command.
  2. If any hardware problems are detected, follow local problem solving procedures.

Error Log Information and TTY Log Identifiers

The following sections discuss important error logging files and commands and common error report messages relating to ttys.

Important Error Logging Files and Commands

Command: **errclear**

This command deletes entries from the error log. The entire log can be erased with `errclear 0` or entries with specified error ID numbers, classes, or types can be removed.

Command: **errpt**

This command generates an error report from entries in the system error log. The most used format for this command is `errpt –a | pg`, which generates a detailed report starting with the most current errors.

File: **/var/adm/ras/errlog**

This file stores instances of errors and failures encountered by system. The **errlog** file tends to become quite lengthy. If not cleared on a regular basis, it can occupy quite a bit of space on your hard disk. Use the **errclear** command mentioned previously to clean out this file.

File: **/usr/include/sys/errids.h**

The **errids.h** header file correlates error IDs with error labels.

Common Error Report Messages

Message	Description	Comments
Core Dump	Software program abnormally terminated	This error is logged when a software program abnormally ends and causes a core dump. Users might not be exiting applications correctly, the system might have been shut down while users were working in application, or the user's terminal might have locked up and the application stopped.
Errlog On	Errdaemon turned on	This error is logged by the **error** daemon when the error logging is started. The system automatically turns off error logging during shutdown.
Lion Box Died	Lost communication with 64–port concentrator	This error is logged by the 64–port concentrator driver if communications with the concentrator are lost. If you receive this error, check the date and time stamp to see if user might have caused this message to occur. A series of these errors can indicate a problem with the 64–port adapter or its associated hardware.
Lion Buffero	Buffer overrun: 64–port concentrator	This error occurs when the hardware buffer in a 64–port concentrator is overrun. If device and cabling allow, try adding request to send (RTS) handshaking to the port and device. Also try lowering the baud rate.

Lion Chunknumc	Bad chunk count: 64–port controller	This error occurs when the value for the number of characters in a chunk does not match the actual values in the buffer. This error may indicate a problem with the hardware; try running diagnostics on devices.
Lion Hrdwre	Cannot access memory on 64–port controller	This error is logged by the 64–port concentrator driver if it is unable to access memory on the 64–port controller.
Lion Mem ADAP	Cannot allocate memory: ADAP structure	This error is logged by the 64–port concentrator driver if the **malloc** routine for the adap structure fails.
Lion Mem List	Cannot allocate memory: TTYP_T List	This error is logged by the 64–port concentrator driver if the **malloc** routine for the *ttyp_t* list structure fails
Lion Pin ADAP	Cannot pin memory: ADAP structure	This error is logged by the 64–port concentrator driver if the **pin** routine for the adap structure fails.
SRC	Software program error	This error is logged by the System Resource Controller (SRC) daemon in the event of some abnormal condition. Abnormal conditions are divided in three areas: failing subsystems, communication failures, and other failures.
Lion Unkchunk	Unknown error code from the 64–port concentrator	Error Code: Number of characters in the chunk received.
TTY Badinput	Bad cable or connection	The port is generating input faster than the system can consume it, and some of that input is being discarded. Usually, the bad input is caused by one or more RS–232 signals changing their status rapidly and repeatedly in a short period of time, causing your system to spend a lot of time in the interrupt handler. The signal errors are usually caused by a loose or broken connector; a bad, ungrounded, or unshielded cable; or by a ”noisy” communications link.

TTY Overrun	Receiver overrun on input	Most TTY ports have a 16–character input FIFO, and the default setting specifies that an interrupt is posted after 14 characters have been received. This error is reported when the driver interrupt handler cleared the input FIFO and data has been lost. Potential solutions depend on the hardware you are using: • 8–port and 128–port adapters Verify that flow control is configured correctly. If it is, run diagnostics, and replace the hardware as appropriate. • Native ports If the problem happens on an idle system, move the workload to a different port. If that corrects the problem, upgrade the system firmware. • General solutions – Reduce the ”RECEIVE trigger level” parameter for this port from 3 to either 2 or 1. – Reduce the line speed on this port. – Examine other devices and processes to try to reduce the time that the system is spending with interrupts disabled.
TTY TTYHOG	TTYHOG overrun	This error is usually caused by a mismatch in the flow control method being used between the transmitter and receiver. The TTY driver has made several attempts to ask the transmitter to pause, but the input has not stopped, causing the data to be discarded. Check the flow control methods configured on each end to make sure that the same method is being used on each.

TTY Parerr	Parity/Framing error on input	This error indicates parity errors on incoming data to asynchronous ports on a character–by–character basis. This is usually caused by a mismatch in line control parameters (parity, line speed, character size, or number of stop bits) between the transmitter and receiver. Line control parameters have to be set the same on both sides in order to communicate.
TTY Prog PTR	Driver internal error	This error is logged by the tty driver if *t_hptr* pointer is null.

Clear a Hung tty Port

In this example, assume that the hung tty port is `tty0`. You must have root authority to be able to complete this procedure.

1. Determine whether the tty is currently handling any processes by typing the following:

   ```
   ps –lt tty0
   ```

   This should return results similar to the following:

   ```
       F S UID   PID  PPID   C PRI NI ADDR    SZ    WCHAN    TTY  TIME CMD
    240001 S 202 22566  3608   0  60 20 781a   444 70201e44   tty0  0:00 ksh
   ```

   The Process ID (PID) here is 22566. To kill this process, type the following:

   ```
   kill 22566
   ```

   Ensure that the process was successfully cleared by typing the command ps –lt tty0. If the process still exists, add the –9 flag to the kill command as shown in the example below.

   Note:: Do not use the –9 option to kill an slattach process. Killing an slattach process with the –9 flag might cause a slip lock to remain in the **/etc/locks** file. Delete this lock file to clean up after slattach.

   ```
   kill –9 22566
   ```

2. Determine if any process is attempting to use the tty by typing the following:

   ```
   ps –ef | grep tty0
   ```

   Note:: If the **ps –ef | grep tty** returns something similar to the following:

   ```
     root 19050      1    0    Mar 06       –  0:00 /usr/sbin/getty
    /dev/tty
   ```

   where the ”–” is displayed between the date (Mar 06) and the time (0:00), this tty does not have the correct cable. This status indicates that the system login process (getty) is attempting to open this tty, and the open process is hanging because the RS–232 signal Data Carrier Detect (DCD) is not asserted. You can fix this by using the correct null modem adapter in the cabling. When getty can open the tty port, the ”–” is replaced by the tty number. For more information on cables, see Attach the Modem with Appropriate Cables.

Note:: The following command can be used to disable the login process on `tty0`.

```
pdisable tty0
```

If the process has been successfully cleared but the tty is still unresponsive, continue to the next step.

3. Type the following command:

   ```
   fuser –k /dev/tty0
   ```

   This will clear any process that can be found running on the port and display the PID. If the tty is still unusable, continue to the next step.

4. Use the **strreset** command to flush outgoing data from the port that is hung due to data that cannot be delivered because the connection to the remote end has been lost.

   Note:: If the **strreset** command fixes the hung port, the port has a cable or configuration problem because the loss of the connection to the remote end should have caused buffered data to be flushed automatically.

   You need to first determine the major and minor device numbers for the tty by typing the following:

   ```
   ls –al /dev/tty0
   ```

   Your results should look similar to the following:

   ```
   crw–rw–rw–    1 root     system    18,   0 Nov  7  06:19 /dev/tty0
   ```

   This indicates that `tty0` has a major device number of 18 and a minor device number of 0. Specify these numbers when using the **strreset** command as follows:

   ```
   /usr/sbin/strreset –M 18 –m 0
    If the tty is still unusable, continue to the next step.
   ```

5. Detach and reattach the cable from the hung tty port. AIX uses the Data Carrier Detect (DCD) signal to determine the presence of a device attached to the port. By dropping DCD, detaching and reattaching the cable will in many cases clear hung processes.

   To determine the location of the port on which the tty is configured, type the following command:

   ```
   lsdev –Cl tty0
   ```

   The results should look similar to the following:

   ```
   tty0    Available   00–00–S1–00  Asynchronous Terminal
   ```

   The third column in the above output indicates the location code of the tty. In this example, `S1` indicates the serial port is configured for native serial port 1. For more information on interpreting location codes, see the Location Codes document in *AIX 5L Version 5.2 System Management Concepts: Operating System and Devices*.

   If the tty is still unusable, continue to the next step.

6. Flush the port using **stty–cxma**. Type the following:

   ```
   /usr/lbin/tty/stty–cxma flush tty0
   ```

   This command is intended for the ttys configured on ports of the 8–port and 128–adapters. In some cases, however, it can be used successfully to flush other tty ports.

   If the tty is still unusable, continue to the next step.

7. On the keyboard of the hung terminal, hold down the `Ctrl` key and press `Q`. This will resume any suspended output by sending an **Xon** character.

   If the tty is still unusable, continue to the next step.

8. A program will sometimes open a tty port, modify some attributes, and close the port without resetting the attributes to their original states. To correct this, bring the tty down to a DEFINED state and then make it available by typing the following:

   ```
   rmdev –l tty0
   ```

   This command leaves the information concerning the tty in the database but makes the tty unavailable on the system.

   The following command reactivates the tty:

   ```
   mkdev –l tty0
   ```

   If the tty is still unusable, consider moving the device to another port and configuring a tty at that location until the system can be rebooted. If rebooting does not clear the port, you most likely have a hardware problem. Check the error report for port hardware problems by entering the following:

   ```
   errpt –a | pg
   ```

   Note:: Some of the preceding commands will not work, and they will give a method error indicating that the device is busy. This is because of the process running on the tty. If none of the steps detailed above free the hung tty, as a last resort, reboot the AIX system and flush the kernel so that the process will go away.

Chapter 8. Data Link Control

Generic data link control (GDLC) is a generic interface definition that allows application and kernel users a common set of commands to control data link control (DLC) device managers within the operating system. For problem determination, see GDLC Problem Determination in *AIX 5L Version 5.2 Communications Programming Concepts*. Topics discussed in this chapter are:

Generic Data Link Control Environment Overview

Generic data link control (GDLC) is a generic interface definition that provides application and kernel users a common set of commands to control DLC device managers within the operating system.

For more information about the GDLC environment, see:

- Implementing the GDLC Interface on page 8-5
- Installing GDLC Data Link Controls on page 8-6
- GDLC Interface ioctl Entry Point Operations on page 8-7
- GDLC Special Kernel Services on page 8-10

The GDLC interface specifies requirements for entry point definitions, functions provided, and data structures for all DLC device managers. DLCs that conform to the GDLC interface include:

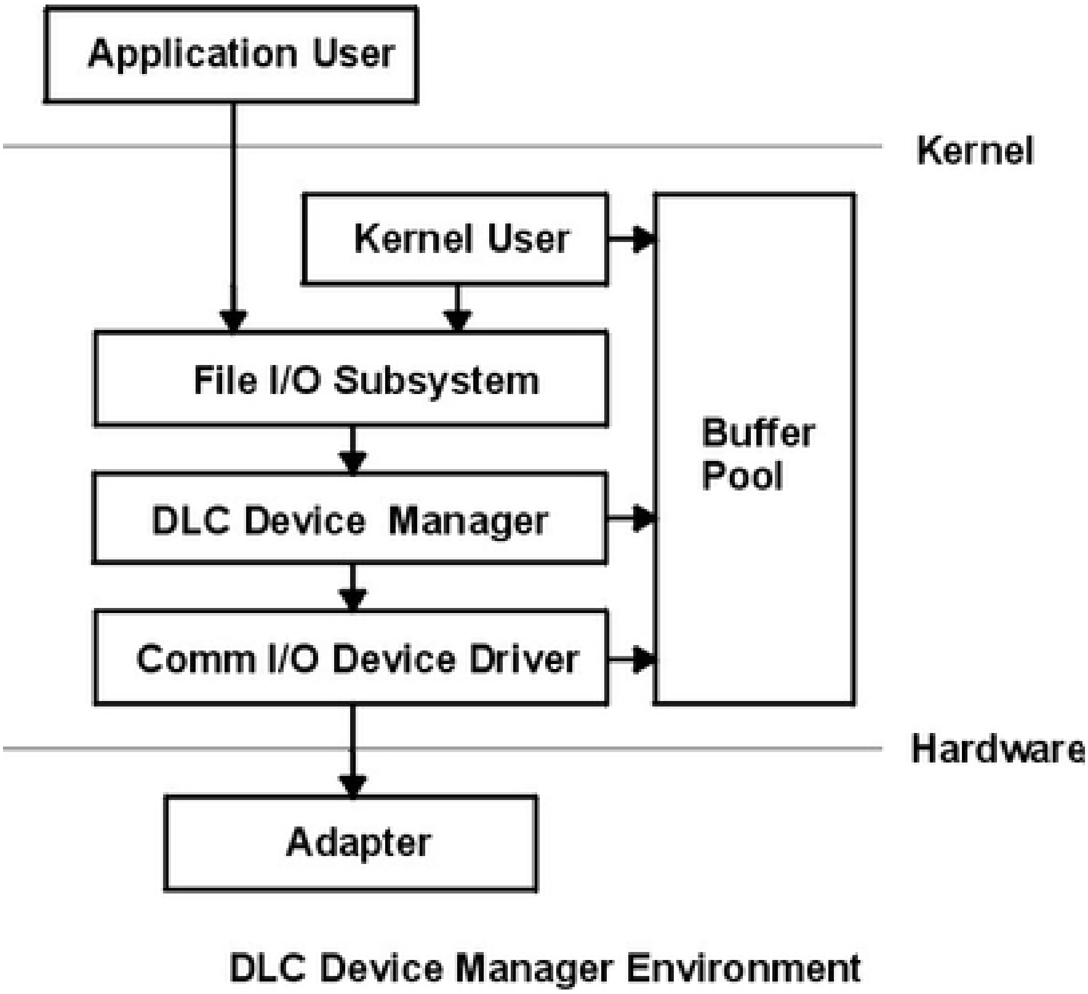
- 8023 (IEEE 802.3 for Ethernet)
- ETHER (Standard Ethernet)
- SDLC (Synchronous Data Link Control)
- TOKEN (Token–Ring)
- FDDI (Fiber Distributed Data Interface)

DLC device managers perform higher–layer protocols and functions beyond the scope of a kernel device driver. However, the managers reside within the kernel for maximum performance and use a kernel device driver for their I/O requests to the adapter. A DLC user is located above or within the kernel.

Synchronous data link control (SDLC) and IEEE 802.2 Data Link Control are examples of DLC device managers. Each DLC device manager operates with a specific device driver or set of device drivers. SDLC, for example, operates with the multiprotocol device driver for the system’s product and its associated adapter.

The basic structure of a DLC environment is shown in the ”DLC Device Manager Environment” figure. Users within the kernel have access to the communications memory buffers (mbufs) and call the add entry points through the **fp** kernel services. Users above the kernel access the standard interface–to–kernel device drivers, and the file system calls the **dd** entry points. Data transfers require a move of data between user and kernel space.

Figure 33. DLC Device Manager Environment This illustration shows the link between the application user and the adapter (hardware level). The areas in between are Kernel User, File I/O Subsystem, DLC Device Manager, Comm I/O Device Driver, and the Buffer Pool. These ”in–between” entities are at the Kernel Level.



The components of the DLC device manager environment are:

Application User Resides above the kernel as an application or access method.

Kernel User Resides within the kernel as a kernel process or device manager.

File I/O Subsystem Converts the file–descriptor and file–pointer subroutines to file pointer accesses of the switch table.

Buffer Pool Provides data–buffer services for the communications subsystem.

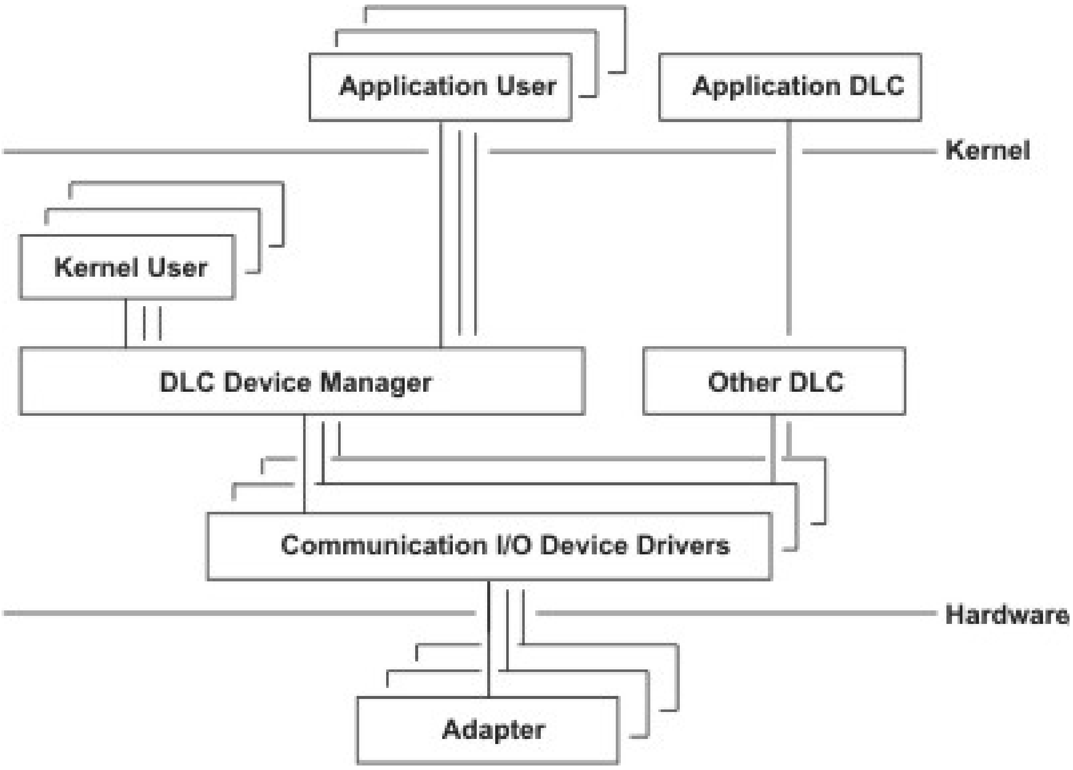
Comm I/O Device Driver Controls hardware adapter I/O and direct memory access (DMA) registers, and routes receive packets to multiple DLCs.

Adapter Attaches to the communications media.

A device manager written in accordance with GDLC specifications runs on all the operating system hardware configurations containing a communications device driver and its target adapter. Each device manager supports multiple users above as well as multiple device drivers and adapters below. In general, users operate concurrently over a single adapter, or each user operates over multiple adapters. DLC device managers vary based on their protocol constraints.

Figure 34 illustrates a multiple user configuration:

Figure 34. Multiple User and Multiple Adapter Configuration This illustration is another view of the kernel level between the application user and the adapter.It shows multiple entities representing multiple users.



Meeting the GDLC Criteria

A GDLC interface must meet the following criteria:

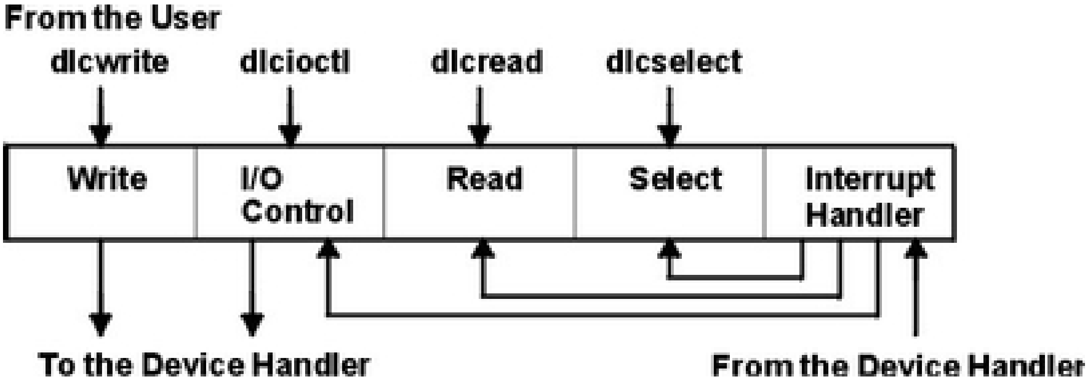
- Be flexible and accessible to both application and kernel users.
- >Have multiple user and multiple adapter capability, allowing protocols to take advantage of multiple sessions and ports.
- Support both connection–oriented and connectionless services where possible.
- Allow transparent data transfer for special requirements beyond the scope of the DLC device manager in use.

Implementing the GDLC Interface

Each DLC device manager is a standard **/dev** entry operating in the kernel as a multiplexed device manager for a specified protocol. For an adapter not in use by DLC, each **open** subroutine to a DLC device manager creates a kernel process. An **open** subroutine is also issued to the target adapter device handler. If needed, issue additional **open** subroutines for multiple DLC adapter ports of the same protocol. Any **open** subroutine targeting the same port does not create additional kernel processes, but links the **open** subroutine with the existing process. There is always one kernel process for each port in use.

The internal structure of a DLC device manager has the same basic structure as a kernel device handler, except that a kernel process replaces the interrupt handler in asynchronous events. The read, write, I/O control, and select blocks function as shown in the ”Standard Kernel Device Manager” figure.

Figure 35. Standard Kernel Device Manager This illustration shows the internal structure of a DLC device manager. This structure consists of a Write, I/O Control, Read, Select, and Interrupt Handler. The Device Manager receives information from the user where it is passed to the various areas before it is passed on to the Device Handler.



Installing GDLC Data Link Controls

You can install DLCs separately or in a group. A DLC device manager is automatically added into the kernel and set to the ”Available” state for each type of DLC installed. Installation can be verified by issuing the **lslpp** command as follows:

```
lslpp –h   dlctype
```

where *dlctype* is one of the following:

bos.dlc.8023	IEEE Ethernet (802.3) Data Link Control
bos.dlc.ether	Standard Ethernet Data Link Control
bos.dlc.fddi	FDDI Data Link Control
bos.dlc.sdlc	SDLC Data Link Control
bos.dlc.token	Token–Ring Data Link Control

Information about an installed DLC can be displayed through Web–based System Manager, the System Management Interface Tool (SMIT), or the command line. On heavily used systems or communications ports, it might be necessary to change the DLC attributes to fine–tune the DLC performance. If receive performance is slow, and the system error log indicates that ring queue overflow is occurring between the DLC and its device handler, increase the DLC queue depth for incoming data. Finally, it is advisable to remove an installed DLC from the kernel when it is not needed for a lengthy period of time. This removal does not remove the DLC from the system, but allows kernel resources to be freed for other tasks until the DLC is needed again. Instructions for all of these tasks are in Managing DLC Device Drivers on page 8-11.

GDLC Interface ioctl Entry Point Operations

The generic data link control (GDLC) interface supports the following **ioctl** subroutine operations:

DLC_ENABLE_SAP Enables a service access point (SAP).

DLC_DISABLE_SAP Disables a SAP.

DLC_START_LS Starts a link station on a particular SAP as a caller or listener.

DLC_HALT_LS Halts a link station.

DLC_TRACE Traces a link station’s activity for short or long activities.

DLC_CONTACT Contacts a remote station for a particular local link station.

DLC_TEST Tests the link to a remote for a particular local link station.

DLC_ALTER Alters a link station’s configuration parameters.

DLC_QUERY_SAP Queries statistics of a particular SAP.

DLC_QUERY_LS Queries statistics of a particular link station.

DLC_ENTER_LBUSY Enters local–busy mode on a particular link station.

DLC_EXIT_LBUSY Exits local–busy mode on a particular link station.

DLC_ENTER_SHOLD Enters short–hold mode on a particular link station.

DLC_EXIT_SHOLD Exits short–hold mode on a particular link station.

DLC_GET_EXCEP Returns asynchronous exception notifications to the application user.

Note:
This **ioctl** subroutine operation is not used by the kernel user since all exception conditions are passed to the kernel user through their exception handler.

DLC_ADD_GRP Adds a group or multicast receive address to a port.

DLC_DEL_GRP Removes a group or multicast receive address from a port.

DLC_ADD_FUNC_ADDR Adds a group or multicast receive functional address to a port.

DLC_DEL_FUNC_ADDR Removes a group or multicast receive functional address from a port.

IOCINFO Returns a structure that describes the GDLC device manager. See the **/usr/include/sys/devinfo.h** file format for more information.

Service Access Point

A SAP identifies a particular user service that sends and receives a specific class of data. This allows different classes of data to be routed separately to their corresponding service handlers. Those DLCs that support multiple concurrent SAPs have addresses known as Destination SAP and Source SAP imbedded in their packet headers. DLCs that can only support a single SAP do not need or use SAP addressing, but still have the concept of enabling the one SAP. In general, there is a SAP enabled for each DLC user on each port.

Most SAP address values are defined by IEEE standardized network–management entities or user–defined values as specified in the *Token–Ring Network Architecture Reference*. Some of the common SAP addresses are:

Null SAP (0x00)	Provides some ability to respond to remote nodes even when no SAP has been enabled. This SAP supports only connectionless service and responds only to exchange identification (XID) and TEST Link Protocol Data Units (LPDUs).
SNA Path Control (0x04)	Denotes the default individual SAP address used by Systems Network Architecture (SNA) nodes.
PC Network NETBIOS (0xF0)	Used for all DLC communication that is driven by Network Basic Input/Output System (NetBIOS) emulation.
Discovery SAP (0xFC)	Used by the local area network (LAN) name–discovery services.
Global SAP (0xFF)	Identifies all active SAPs.

Link Station

A link station (LS) identifies an attachment between two nodes for a particular SAP pair. This attachment can operate as a connectionless service (datagram) or connection–oriented service (fully sequenced data transfer with error recovery). In general, there is one LS started for each remote attachment.

Local–Busy Mode

When an LS is operating in a connection–oriented mode, it needs to stop the remote station's sending of information packets for reasons such as resource outage. Notification can then be sent to the remote station to cause the local station to enter local–busy mode. Once resources are available, the local station notifies the remote that it is no longer busy and that information packets can flow again. Only sequenced information packets are halted with local–busy mode. All other types of data are unaffected.

Short–Hold Mode

Use the short–hold mode of operation when operating over data networks with the following characteristics:

- Short call–setup time
- Tariff structure that specifies a relatively small fee for the call setup compared to the charge for connect time.

During short–hold mode, an attachment between two stations is maintained only while there is data available for transfer between the two stations. When there is no data to send, the attachment is cleared after a specified time–out period and only reestablished when there is new data to transfer.

Testing and Tracing a Link

To test an attachment between two stations, instruct an LS to send a test packet from the local station. This packet is echoed back from the remote station if the attachment is operating correctly.

Some data links are limited in their support of this function due to protocol constraints. SDLC, for example, only generates the test packet from the host or primary station. Most other protocols, however, allow test packets to be initiated from either station.

To trace a link, line data, and special events (such as station activation, termination, and time outs), obtain a generic trace channel and instruct an LS to write its trace logs into the generic trace facility for each LS. This function helps determine the cause of certain communications attachment problems. Both short and long trace entries are supported.

Statistics

Both SAP and LS statistics can be queried by a GDLC user. The statistics for a SAP consist of the current SAP state and information about the device handler. LS statistics consist of the current station states and various reliability, availability, and serviceability counters that monitor the activity of the station from the time it is started.

GDLC Special Kernel Services

Generic data link control (GDLC) provides special services for a kernel user. However, a trusted environment must exist within the kernel. Instead of the DLC device manager copying asynchronous event data into user space, the kernel user must specify function pointers to special routines called function handlers. Function handlers are called by the DLC at the time of execution. This allows maximum performance between the kernel user and the DLC layers. Each kernel user is required to restrict the number of function handlers to a minimum path length and use the communications memory buffer (mbuf) scheme.

A function handler must never call another DLC entry directly. This is because direct calls are made under lock, causing a fatal sleep. The only exception to this rule is that a kernel user might call the **dlcwritex** entry point during its service of any of the four receive data functions. Calling the **dlcwritex** entry point allows immediate responses to be generated without an intermediate task switch. Special logic is required within the DLC device manager to check the process identification of the user calling a write operation. If it is a DLC process and the internal queuing capability of the DLC has been exceeded, the write is sent back with a bad return code (**EAGAIN** return value) instead of putting the calling process (DLC) to sleep. It is then up to the calling user subroutine to return a special notification to the DLC from its receive data function to ensure a retry of the receive buffer at a later time.

The user–provided function handlers are:

Datagram Data Received Routine
Called any time a datagram packet is received for the kernel user.

Exception Condition Routine
Called any time an asynchronous event occurs that must notify the kernel user, such as SAP Closed or Station Contacted.

I–Frame Data Received Routine
Called each time a normal sequenced data packet is received for the kernel user.

Network Data Received Routine
Called any time network–specific data is received for the kernel user.

XID Data Received Routine
Called any time an exchange identification (XID) packet is received for the kernel user.

The **dlcread** and **dlcselect** entry points for DLC are not called by the kernel user because the asynchronous functional entries are called directly by the DLC device manager. Generally, any queuing of these events must occur in the user's function handler. If, however, the kernel user cannot handle a particular receive packet, the DLC device manager may hold the last receive buffer and enter one of two special user–busy modes:

User–Terminated Busy Mode (I–frame only)

If the kernel user cannot handle a received I–frame (due to problems such as queue blockage), a DLC_FUNC_BUSY return code is given back, and DLC holds the buffer pointer and enters local–busy mode to stop the remote station's I–frame transmissions. The kernel user must call the Exit Local Busy function to reset local–busy mode and start the reception of I–frames again. Only normal sequenced I–frames can be stopped. XID, datagram, and network data are not affected by local–busy mode.

Timer–Terminated Busy Mode (all frame types)

If the kernel user cannot handle a particular receive packet and wants DLC to hold the receive buffer for a short period and then re–call the user receive function, a DLC_FUNC_RETRY return code is given back to DLC. If the receive packet is a sequenced I–frame, the station enters local–busy mode for that period. In all cases, a timer is started; once the timer expires, the receive data functional entry is called again.

Managing DLC Device Drivers

A DLC must be added to the system prior to use. Each installed DLC is automatically added after installation and at each system restart (see Installing GDLC Data Link Controls on page 8-6). If a DLC has been removed without a subsequent restart, it can be re–added.

Tasks for managing DLC device drivers

Task	**SMIT fast path**	**Command or file**	Web-based System Manager Management Environment [7]
Add an Installed DLC	Choose one (by device driver name): **smit cmddlc_sdlc** **smit cmddlc_token** **smit cmddlc_qllc** **smit cmddlc_ether** [1] **smit cmddlc_fddi** then select **Add**	**mkdev** [2]	
Change DLC Attributes [3,4]	Choose one (by device driver name): **smit cmddlc_sdlc_ls** **smit cmddlc_token_ls** **smit cmddlc_qllc_ls** **smit cmddlc_ether_ls** [1] **smit cmddlc_fddi_ls**	**chdev** [2]	
Start DLC Local Area Network Monitor Trace [5]	**smit trace**	**trace –j *nnn*** where the value *nnn* is the hook ID to be traced	
Stop DLC Local Area Network Monitor Trace	**smit trcstop**	**trcstop** [2]	
Generate DLC Local Area Network Monitor Trace Report	**smit trcrpt**	**trcrpt –d *nnn*** where the value *nnn* is the hook ID to be reported	

List Current DLC Information [3]	Choose one (by device driver name): **smit cmddlc_sdlc_ls** **smit cmddlc_token_ls** **smit cmddlc_qllc_ls** **smit cmddlc_ether_ls** [1] **smit cmddlc_fddi_ls**	**lsdev** [2] or **lsattr** @T>2	
Remove a DLC [3,6]	Choose one (by device driver name): **smit cmddlc_sdlc_rm** **smit cmddlc_token_rm** **smit cmddlc_qllc_rm** **smit cmddlc_ether_rm** [1] **smit cmddlc_fddi_rm**	**rmdev** [2]	

Notes:

1. The SMIT fast path for an Ethernet device manager includes both Standard Ethernet and IEEE 802.3 Ethernet device managers.
2. Details about command line options are provided in the command descriptions for **mkdev**, **chdev**, **trace**, **trcstop**, **trcrpt**, **lsdev**, **lsattr**, or **rmdev** in *AIX 5L Version 5.2 Commands Reference*.
3. A DLC must be installed and added before you can list, show, change, or remove its attributes (see Installing GDLC Data Link Controls on page 8-6). An attribute change is only successful if there are no active opens against the target DLC. Before issuing the change action, the user might have to stop services such as SNA, OSI, or NetBIOS from using the DLC.
4. Changing the receive–queue size directly affects system resources. Make this change only if the DLC is having receive–queue problems, such as sluggish performance or overflows between the DLC and its device handler.
5. Exercise caution when enabling the monitor trace since it directly affects the performance of the DLCs and their associates.
6. Removing a DLC is only successful if there are no active opens against the target DLC. Before issuing the remove action, the user may have to stop services such as SNA, OSI, or NetBIOS from using the DLC.
7. These tasks are not available in Web-based System Manager Management Environment.

Chapter 9. Basic Networking Utilities

This chapter presents information on installing, configuring, and maintaining Basic Network Utilities (BNU). The following topics are discussed:

- BNU Overview on page 9-2
- Configuring BNU on page 9-10
- Maintaining BNU on page 9-17
- BNU Configuration Files on page 9-30
- BNU Files, Commands, and Directories Reference on page 9-38.

BNU Overview

The Basic Networking Utilities (BNU) are a group of programs, directories, and files that can be used to communicate with any UNIX system on which a version of the UNIX–to–UNIX Copy Program (UUCP) is running. BNU is one of the Extended Services programs that can be installed with the base operating system.

A group of commands related to UUCP, a UNIX–to–UNIX communication program developed by AT&T and modified as part of the Berkeley Software Distribution (BSD) are contained in BNU. BNU provides commands, processes, and a supporting database for connections to local and remote systems. Communication networks such as Token–Ring and Ethernet are used to connect systems on local networks. A local network can be connected to a remote system by hardwire or telephone modem. Commands and files can then be exchanged between the local network and the remote system.

Topics discussed in this section are:

- How BNU Works on page 9-2
- BNU File and Directory Structure on page 9-3
- BNU Security on page 9-5
- BNU Daemons on page 9-7

Before users on your system can run BNU programs, BNU must be installed and configured.

BNU is controlled by a set of configuration files that determine whether remote systems can log in to the local system and what they can do after they log in. These configuration files must be set up according to the requirements and resources of your system.

To maintain BNU, you must read and remove log files periodically and check the BNU queues to ensure jobs are correctly transferring to remote systems. You must also periodically update the configuration files to reflect changes in your system or remote systems.

For more information, see:

- Configuring BNU on page 9-10
  - Information to Collect before Configuring BNU on page 9-10
- Maintaining BNU on page 9-17
  - Working with BNU Log Files on page 9-17
  - BNU Maintenance Commands on page 9-18.

How BNU Works

BNU uses a set of hardware connections and software programs to communicate between systems. A structure of directories and files tracks BNU activities. This structure includes a set of public directories, a group of administrative directories and files, configuration files, and lock files. Most of the directories for BNU are created during the installation process. Some of the administrative directories and files are created by various BNU programs.

With the exception of the remote login commands, BNU works as a batch system. When a user requests a job sent to a remote system, BNU stores the information needed to complete the job. This is known as *queuing* the job. At scheduled times, or when a user instructs it to do so, BNU contacts various remote systems, transfers queued work, and accepts jobs. These transfers are controlled by the configuration files on your system and those of the remote system.

National Language Support for BNU Commands

All BNU commands, except **uucpadm**, are available for National Language Support. User names need not be in ASCII characters. However, all system names must be in ASCII characters. If a user attempts to schedule a transfer or a remote command execution involving non–ASCII system names, BNU returns an error message.

BNU File and Directory Structure

BNU uses a structure of directories and files to keep track of activities. This structure includes the following:

- BNU Public Directories
- BNU Configuration Files
- BNU Administrative Directories and Files
- BNU Lock Files.

Most of the directories for BNU are created during the installation process. Some of the administrative directories and files are created by various BNU programs as they run.

BNU Public Directories

When specified, the BNU public directory (**/var/spool/uucppublic**) stores files that have been transferred to the local system from other systems. The files wait in the public directory until users claim them. The public directory is created when BNU is installed. Within the public directory, BNU creates a subdirectory for each remote system that sends files to the local system.

BNU Configuration Files

The BNU configuration files, also known as the BNU supporting database, reside in the **/etc/uucp** directory. The files must be configured specifically for your system. They are owned by the uucp login ID and can be edited only with root authority. The configuration files contain information about:

- Accessible remote systems
- Devices for contacting the remote systems
- Times to contact the remote systems
- What the remote systems are allowed to do on your system.

Some configuration files also specify limits on BNU activities that prevent your system from becoming overloaded.

The BNU configuration files include:

Devices	Contains information about available devices, including both modems and direct connections.
Dialcodes	Contains dialing code abbreviations, which allow you to shorten phone numbers in the **Systems** file.
Dialers	Specifies calling command syntax for a specific modem type (”dialer”).
Maxuuscheds	Limits simultaneous scheduled jobs.
Maxuuxqts	Limits simultaneous remote command executions.
Permissions	Contains access permission codes. This file is the primary file for determining the security for BNU.
Poll	Specifies when the BNU program should poll remote systems to initiate tasks.

Sysfiles Lists files that serve as the **Systems**, **Devices**, and **Dialers** files for the BNU configuration. If this file is not used, the default files are **/etc/uucp/Systems**, **/etc/uucp/Devices**, and **/etc/uucp/Dialers**.

Systems Lists accessible remote systems and information needed to contact them, including the device to use and the user name and password combinations you need to log in. Also specifies the times when the systems can be contacted.

The configuration files cross–reference each other when BNU is in use. For example:

- The **Devices** file contains a *Token* field that refers to entries in the **Dialers** file.
- The **Systems** file contains an entry for a *Class* of device. A device of each *Class* referred to in the **Systems** file must be defined in the **Devices** file.
- The **Poll** file contains entries for systems your system calls. Each of these systems must be defined in the **Systems** file.

Entries in the BNU configuration files depend on the types of connections between your system and each remote system. For example, special entries must be made if Transmission Control Protocol/Internet Protocol (TCP/IP) or direct connections are used to contact other systems. If modems are used to contact other systems, the modems must be defined in the **Dialers** file.

The **Systems**, **Devices**, and **Permissions** files must be configured on your system before you can contact remote systems using BNU. Other configuration files enable you to use BNU capabilities, such as automatic polling. Many of the configuration files must be modified periodically to reflect changes to your system or the systems you contact. The **Sysfiles** file can be used to specify files other than the default **Systems**, **Devices**, and **Dialers** files to fulfill the same role.

BNU Administrative Directories and Files

The BNU administrative directories and files are in subdirectories of the **/var/spool/uucp** directory. These directories and files contain two types of information:

- Data waiting to be transferred to other systems
- Log and error information about BNU activities.

Under the **/var/spool/uucp** directory, BNU creates the following directories:

.Admin Contains four administrative files:

- **audit**
- **Foreign**
- **errors**
- **xferstats**

These files contain error and log information about BNU activities.

.Corrupt Contains copies of files that cannot be processed by the BNU program.

.Log and **.Old** Contain log files from BNU transactions.

.Status Stores the last time the **uucico** daemon tried to contact remote systems.

.Workspace Holds temporary files that the file transport programs use internally.

.Xqtdir Contains execute files with lists of commands that remote systems can run.

SystemName Contains files used by file transport programs. These files are:

- Command (**C.***)
- Data (**D.***)
- Execute (**X.***)
- Temporary (**TM.***)

BNU creates a *SystemName* directory for each remote system it contacts.

The directories whose names begin with a dot are *hidden*. They cannot be found with an **ls** or **li** command unless the **–a** flag is used. When the **uucico** daemon is started, it searches the **/var/spool/uucp** directory for work files and transfers the files from any directory that is not hidden. The **uucico** daemon sees only the *SystemName* directories, not the other administrative directories.

The files in the hidden directories are owned by the uucp login ID. These files can be accessed only with root authority or with a login ID with a UID of 5.

For further information about maintaining the BNU administrative directories, see Maintaining BNU on page 9-17.

BNU Lock Files

The BNU lock files are stored in the **/var/locks** directory. When BNU uses a device to connect to a remote computer, it places a lock file for that device in the **/var/locks** directory. When another BNU program or any other program needs the device, that program checks the **/var/locks** directory for a lock file. If a lock file exists, the program waits until the device is available or uses another device for the communication.

In addition, the **uucico** daemon places lock files for remote systems in the **/var/locks** directory. Before contacting a remote system, the **uucico** daemon checks the **/var/locks** directory for a lock file for that system. These files prevent other instances of the **uucico** daemon from establishing duplicate connections to the same remote system.

Note:: Other software besides BNU, such as Asynchronous Terminal Emulation (ATE) and TCP/IP, uses the **/var/locks** directory.

BNU Security

Because other systems contact your system to log in, transfer files, and enter commands, BNU provides a means to establish security. BNU security enables you to restrict what users of remote systems can do on the local system (users of remote systems can also restrict what you can do on their systems). BNU runs several daemons to complete its activities and uses administrative directories to store the files it needs. BNU also keeps a log of its own activities.

BNU security works on several levels. When you configure BNU, you can determine:

- Who on your system has access to BNU files.
- Which remote systems your system can contact.
- How users on remote systems log in to your system.
- What users on remote systems can do on your system once they log in.

uucp Login ID

When BNU is installed, all of the configuration files, daemons, and many of the commands and shell procedures are owned by the uucp login ID. The uucp login ID has a user ID (UID) of 5 and a group ID (GID) of 5. The **cron** daemon reads the **/var/spool/cron/crontabs/uucp** file to schedule automatic jobs for BNU.

Usually, logging in as user uucp is not allowed. To change files that are owned by the uucp login ID, log in with root authority. **Attention:** Allowing remote systems to log in to the local system with the uucp login ID seriously jeopardizes the security of the local system. Remote systems logged in with the uucp ID can display and possibly modify the local **Systems** and **Permissions** files depending on the other permissions specified in the LOGNAME entry. It is strongly recommended that you create other BNU login IDs for remote systems and reserve the uucp login ID for the person responsible for administering BNU on the local system. For the best security, each remote system that contacts the local system must have a unique login ID with a unique UID number.

The operating system provides a default nuucp login ID for transferring files.

BNU Login IDs

The startup shell for BNU login IDs is the **uucico** daemon (**/usr/sbin/uucp/uucico**). When remote systems call your system, they automatically start the **uucico** daemon on your system. Login IDs for BNU have a uucp group ID of 5.

Login IDs used by remote systems need passwords. In order to prevent security from prompting a new BNU login ID for a new password when the remote system logs in, you must set the password as soon as you create the account. To do this, use the **passwd** command followed by the **pwdadm** command. For example, to set a password for the login ID `nuucp`, log in as the root user and enter the following commands:

```
passwd nuucp
```

```
pwadm –f NOCHECK
nuucp
```

The system prompts you for a password for the `nuucp` login ID. Completing these steps allows the remote system to log in without being immediately prompted for a new password (which the batch–oriented `nuucp` login ID can not provide).

After creating the login ID for a remote system, notify that system BNU administrator of the login ID and password to access your system.

Creating a BNU Administrative Login ID

A user with root authority can set up a BNU administrative login ID. This is useful if you want to delegate BNU administration duties to a user without root authority. The BNU administrative login ID should have password security, a UID of 5, and be in a uucp group ID 5. The login shell for the administrative login should be the **/usr/bin/sh** program (instead of the **uucico** daemon). Giving the BNU administrative login a UID of 5 causes it to have the same privileges as the **uucp** login ID. Thus, for security, remote systems should not be allowed to log in as the BNU administrator.

Security and the Systems and remote.unknown Files

On most BNU systems, only remote systems listed in the **/etc/uucp/Systems** file or one of its substitutes (specified in the **Sysfiles** file) can log in to the local system. The **/usr/sbin/uucp/remote.unknown** script is executed whenever an unknown system attempts to call the local system. This script refuses to let the unknown system log in and makes an entry in the **/var/spool/uucp/.Admin/Foreign** file recording the time of the login attempt.

With root authority, or as a BNU administrator, you can modify the **remote.unknown** shell procedure to log more information about the remote system or to store the information in a different file. For example, you can modify the shell procedure to send mail to the BNU administrator whenever an unknown system tries to log in.

By taking away execute permissions on the **remote.unknown** shell procedure, you enable unknown machines to log in. In this case, you should add a MACHINE=OTHER entry to the **/etc/uucp/Permissions** file to establish permissions for the unknown machines.

Your system can contact only remote systems listed in the **Systems** file. This prevents users on your system from contacting unknown systems.

Security and the Permissions File

The **/etc/uucp/Permissions** file determines:

- Remote login user names for logging in to the local system
- Approved commands and privileges for remote systems logging in to the local system.

The **/etc/uucp/Permissions** file contains two types of entries:

LOGNAME Defines login names and the privileges associated with them. LOGNAME entries take effect when a remote system calls the local system and attempts to log in.

MACHINE Defines machine names and the privileges associated with them. MACHINE entries take effect when the remote system attempts to carry out commands on the local system.

Options in the **Permissions** file enable you to establish various levels of security for each remote system. For example, if many remote systems share one login ID on the local system, use the VALIDATE option to require each remote system to use a unique login ID. The SENDFILES, REQUEST, and CALLBACK options specify which system has control, keeping the local system in control of transactions if necessary.

The READ, WRITE, NOREAD, and NOWRITE options define access to specific directories on the local system. These options also control where on your system remote users can place data. The COMMANDS option limits the number of commands users on remote systems can execute on the local system. The COMMANDS=ALL option allows total privileges to systems closely associated with your system. **Attention:** The COMMANDS=ALL option can seriously jeopardize the security of your system.

BNU Daemons

The BNU software includes four daemons stored in the **/usr/sbin/uucp** directory:

uucico Facilitates file transfers (see Using the uucico Daemon)

uusched Facilitates work request scheduling of files queued in the local spooling directory (see Using the uusched Daemon)

uuxqt Facilitates remote command executions (see Using the uuxqt Daemon)

uucpd Facilitates communications using TCP/IP (see Using the uucpd Daemon)

The **uucico**, **uusched**, and **uuxqt** daemons are started by the **cron** daemon according to a schedule set by the BNU administrator. With root authority, you can also start these daemons manually. The **uucpd** daemon should be started by the TCP/IP **inetd** daemon.

Using the uucico Daemon

The **uucico** daemon transports the files required to send data from one system to another. The **uucp** and **uux** commands start the **uucico** daemon to transfer command, data, and execute files to the designated system. The **uucico** daemon is also started periodically by the BNU scheduler, the **uusched** daemon. When started by the **uusched** daemon, the **uucico** daemon attempts to contact other systems and execute the instructions in the command files.

How the Daemon Process Begins

To run the instructions in the command files, the **uucico** daemon first checks the **/etc/uucp/Systems** file (or one or more other files specified by **/etc/uucp/Sysfiles**) for the system to be called. The daemon then checks the **Systems** file entry for a valid time to call. If the time is valid, the **uucico** daemon checks the *Type* and *Class* fields and accesses the **/etc/uucp/Devices** file (or one or more other files specified by **/etc/uucp/Sysfiles**) for a device that matches.

After finding a device, the **uucico** daemon checks the **/var/locks** directory for a lock file for the device. If one exists, the daemon checks for another device of the requested type and speed.

When no device is available, the daemon returns to the **Systems** files for another entry for the remote system. If one exists, the daemon repeats the process of searching for a device. If another entry is not found, the daemon makes an entry in the **/var/spool/uucp/.Status/** *SystemName* file for that remote system and goes on to the next request. The command file remains in the queue. The **uucico** daemon attempts the transfer again at a later time. The later attempt is called a *retry*.

When the Daemon Reaches the Remote System

When the **uucico** daemon reaches the remote system, it uses the instructions in the **Systems** files to log in. This causes an instance of the **uucico** daemon to be invoked on the remote system as well.

The two **uucico** daemons, one on each system, work together to make the transfer. The **uucico** daemon on the calling system controls the link, specifying the requests to be performed. The **uucico** daemon on the remote system checks the local permissions for whether they allow the request to be performed. If so, the file transfer starts.

After the **uucico** daemon on the calling system has finished transferring all requests it has for the remote system, it sends a hangup request. When the remote **uucico** daemon has transactions to send to the calling system, it denies the hangup request, and the two daemons reverse roles.

Note:: Either the **/etc/uucp/Permissions** file on the local system or the **/etc/uucp/Permissions** file on the remote system can forbid the daemons to reverse roles. In this case, the remote system must wait to transfer files until it calls the local system.

When nothing is left to be transferred in either direction, the two **uucico** daemons hang up. At this point, the **uuxqt** daemon (Using the uuxqt Daemon on page 9-9) is called to execute remote command requests.

Throughout the transfer process, the **uucico** daemons on both systems log messages in the BNU log and error files.

Using the uusched Daemon

The **uusched** daemon schedules the transfer of files that are queued in the spooling directory on the local system. The spooling directory is **/var/spool/uucppublic**. When the **uusched** daemon is invoked, it scans the spooling directory for command files, then randomizes the files and starts the **uucico** daemon. The **uucico** daemon transfers the files.

Using the uuxqt Daemon

When a user issues the **uux** command to run a specified command on a designated system, the **uuxqt** daemon runs the command. After creating the necessary files, the **uux** command starts the **uucico** daemon, which transfers those files to the public spooling directory on the specified system.

The **uuxqt** daemon periodically searches the spooling directory for command–execution requests on every connected system. When it locates such a request, the **uuxqt** daemon checks for necessary files and permissions. Then, if permitted, the daemon runs the specified command.

Using the uucpd Daemon

The **uucpd** daemon must be able to run on the remote system before BNU can establish communications with a remote computer with Transmission Control Protocol/Internet Protocol (TCP/IP). The **uucpd** daemon is a subserver of the TCP/IP **inetd** daemon and is started by the **inetd** daemon.

By default, the **uucpd** daemon is commented in the **inetd.conf** file. To use it, you must remove the comment character and restart **inetd**. However, if this has been changed on your system, you might need to reconfigure the **inetd** daemon to start the **uucpd** daemon.

Configuring BNU

The following procedures detail how to configure Basic Network Utilities (BNU) for various types of connections, including hardwired, modem, and Transmission Control Protocol/Internet Protocol (TCP/IP) connections.

Prerequisites

- BNU must be installed on your system.
- You must have root user authority to edit the BNU configuration files.
- If you are using direct connections for BNU communications, the appropriate hardwired connections between your system and the remote systems must be set up.
- If you are using modems for BNU communications, you must have installed and configured each modem.
- If one or more of your connections uses TCP/IP, then TCP/IP must be running between your system and the apropriate remote systems.
- Collect the information you need to configure BNU (see Information to Collect before Configuring BNU). This information should include a list of remote systems and lists of devices and modems to use to connect to the systems.

Information to Collect before Configuring BNU

Before configuring BNU, gather the information listed:

- For each remote system your system will call, collect the following information:
  - System name
  - Login name your system should use on the remote system
  - Password for the login name
  - Login and password prompts on the remote system
  - Type of connection you will use to reach the remote system (TCP/IP, direct, or telephone).
- If the connection is direct, collect:
  - The bit rate of the connection
  - The port on the local system to which the connection is attached.
- If the connection is a telephone connection, collect:
  - The telephone number of the remote system
  - The speed of your modem that is compatible with that of the remote system.

Note:: If any of the remote systems will call your system, ensure the BNU administrator on each of the remote systems has all the preceding information about your system.

- For each local modem that you will use for BNU connections, collect the following information:
  - The chat script for the modem (consult the modem documentation)

    Note:: For some modems, the chat script is already in the **/etc/uucp/Dialers** file.
  - The local port for the modem.

Using the information you collect, make a list of each device you need to connect to a remote system. Following is a sample list for local system `morgan`:

```
direct:
hera 9600 tty5
zeus& 2400 tty2
ariadne 2400 tty1
hayes modem (tty3):  apollo, athena
TCP/IP:  merlin, arthur, percy
```

To connect to system `hera`, a `direct` connection at a speed of `9600` from port `tty5` is used. To connect to system `apollo`, the `hayes` modem connected to port `tty3` is used. TCP/IP is used to connect to systems `merlin`, `arthur`, and `percy`.

Procedure

For BNU to function correctly at your site, you must configure the remote communications facilities to:

- List the devices used to establish a hardwired, telephone, or modem communications link.
- List the modems used to contact remote systems over the telephone network.
- List the accessible remote systems.
- List the alphabetic abbreviations representing the prefixes of telephone numbers used to contact the specified remote systems (optional).
- Set access permissions specifying the ways in which local and remote systems may communicate.
- Schedule monitoring for the networked remote systems (optional).

To create these lists, permissions, schedules, and procedures:

- Modify the BNU configuration files.
- Edit the **/var/spool/cron/crontabs/uucp** file to remove the comment characters (#) from the beginnings of the lines that schedule the automatic maintenance routines.

You must configure the **Systems**, **Devices**, and **Permissions** files before BNU will run correctly at your site. However, it is not necessary to modify the BNU configuration files in any particular order.

To configure BNU on your system:

1. Make sure that BNU is installed on your system by running the command:

   ```
   lslpp –h bos.net.uucp
   ```

   If BNU is installed, you will see `bos.net.uucp` in the output. If you do not see it, install it from the install tape.

2. Set up appropriate login IDs and passwords for remote systems that will call your system, and tell the person responsible for administering BNU or UNIX–to–UNIX Copy Program (UUCP) on each remote system the login and password you have provided. This is done by editing the **/etc/passwd**, **/etc/group**, **/etc/security/login.cfg**, and **/etc/security/passwd** files. **Attention:** Allowing remote systems to log into the local system with the UUCP login ID seriously jeopardizes the security of your system. Remote systems logged in with the UUCP ID can display and possibly modify (depending on the permissions specified in the LOGNAME entry of the **Permissions** file) the local **Systems** and **Permissions** files. It is strongly recommended that you create other BNU login IDs for remote systems and reserve the uucp login ID for the person administering BNU on the local system. For the best security, each remote system that contacts the local system should a have unique login ID with a unique UID number. These login IDs should have GIDs of 5. By default, the operating system includes the nuucp login ID for transferring files.

a. You have the option of maintaining separate logins or having one login for all BNU connections. If you need to maintain complete control over access by each individual machine, you must create separate login IDs, as well as combine the MACHINE and LOGNAME entries in the **Permissions** file. A few example **/etc/passwd** entries are shown here:

```
Umicrtk:!:105:5:micrtk
uucp:/usr/spool/uucppublic:/usr/sbin/uucp/uucico
 Ufloydl:!:106:5:floydl
uucp:/usr/spool/uucppublic:/usr/sbin/uucp/uucico
 Uicus:!:107:5:icus uucp:/usr/spool/uucppublic:/usr/sbin/uucp/uucico
 Urisctkr:!:108:5::/usr/spool/uucppublic:/usr/sbin/uucp/uucico
```

b. If you want to have one set of permissions and do not want to maintain separate control for any of your UUCP connections, you can have a single login for all machines such as the following:

```
nuucp:!:6:5::/usr/spool/uucppublic:/usr/sbin/uucp/uucico
```

c. The user ID (the third colon separated field) must be unique to avoid a security risk. The group ID (the fourth separated field) must be 5, the same group as uucp. You can change the home directory (the sixth field) to any valid directory, but the login shell (the seventh field) must be `/usr/sbin/uucp/uucico`.

d. Make sure that the **/etc/group** file contains the new users. An example of such an entry is:

```
uucp:!:5:uucp,uucpadm,nuucp,Umicrtk,Uicus,Urisctakr
```

e. You may want to add any users to group uucp who will be using modems to dial out with programs other than the **cu** command.

f. After editing these files as root, set up a password for the new users with the command `passwd UserName`.

g. Sometimes, the default herald with all of its Ctrl–J's, will cause a uucico process that is trying to login to give up. (You may see the message `Enough already`.) You can avoid that by commenting out (with asterisks) the default stanza, and defining a stanza for your tty something like this:

```
 /dev/tty0:
              herald = ″\nrisc001 login:″
```

h. If you change a password from the root login, the flags entry in the stanza for the user in `/etc/security/passwd` will contain the following:

```
flags = ADMCHG
```

Change it to:

```
flags =
```

Otherwise, when the remote uucico logs in, it will be prompted to enter a new password, which it cannot do. Hence the login will fail.

i. Using an ASCII text editor or the **uucpadm** command, edit the **Poll** file. Add an entry for each system your system will poll.

Note:: The systems listed in the **Poll** file must also be listed in the **/etc/uucp/Systems** file.

j. Using an ASCII text editor, edit the **/var/spool/cron/crontabs/uucp** file. Remove the comment characters (#) from the lines that run the **uudemon.hour** and **uudemon.poll** commands. You can change the times these commands are run. However, be sure to schedule the **uudemon.poll** command approximately five minutes *before* you schedule the **uudemon.hour** command.

k. Check to make sure your changes took effect by running this command:

```
crontab –l
```

l. Set up the ‘BNU data files: Systems, Permissions, Devices, Dialers, and Sysfiles. You could use the **/usr/sbin/uucp/uucpadm** command to initially set up the files and then edit them to suit your exact needs. Note that the Sysfiles file allows you to specify files other than **/etc/uucp/Systems**, **/etc/uucp/Devices**, and **/etc/uucp/Dialers** for BNU configuration. See **Sysfiles** for more information.

3. Decide whether to use dial–code abbreviations for telephone numbers (see the **Dialcodes** file format). If you decide to use dial–code abbreviations in the **Systems** files, set up the **Dialcodes** entry for each abbreviation. Refer to Dialcodes File Format for BNU in *AIX 5L Version 5.2 Files Reference* for details.

   If you are using TCP/IP for your BNU connections, use the **netstat** command to see whether the **uucpd** daemon is runnable, by entering:

   ```
   netstat –a
   ```

   The **uucpd** daemon is started by the **inetd** daemon. If the **uucpd** daemon is not able to run, reconfigure the **inetd** daemon to start the **uucpd** daemon (see Configuring the inetd Daemon).

4. Using the list of devices you collected before beginning this procedure, modify the **Devices** file on your system. Make an entry for each modem and each direct connection (see Information to Collect before Configuring BNU). If you are using TCP/IP, make sure you uncomment the TCP/IP entry in the **Devices** file. You can configure the **/etc/uucp/Sysfiles** file to specify other files to use for Devices configuration. Refer to the Devices File Format for BNU in *AIX 5L Version 5.2 Files Reference* for details on the Devices file. Refer to Sysfiles File Format for BNU for details on the Sysfiles file in *AIX 5L Version 5.2 Files Reference*.

   Also, if you are using TCP/IP, check to see whether the **/etc/services** file includes:

   ```
   uucp        540/tcp         uucpd
   ```

   If not, add the line.

5. Using your information about each remote system that you collected before beginning this procedure, modify the **Systems** file on your system (see Information to Collect before Configuring BNU). Use the commented examples in the **Systems** file as a guide when specifying your configuration. See the ”BNU Systems File Format” in *AIX 5L Version 5.2 Files Reference* for details. If you are using TCP/IP, ensure the host–name table in the **/etc/hosts** file includes the name of the remote computer with which you want to connect. You can configure the **/etc/uucp/Sysfiles** file to specify other files to use for Systems configuration. Refer to Sysfiles File Format for BNU in *AIX 5L Version 5.2 Files Reference* for more information.

6. Using the information about devices and modems that you collected before beginning this procedure, make sure the **Dialers** file on your system contains an entry for each modem (see Information to Collect before Configuring BNU). If you are using TCP/IP and direct connections, make sure the TCP/IP entry and direct entries are present in the file. Refer to Dialers File Format for BNU in *AIX 5L Version 5.2 Files Reference* for details. You can configure the **/etc/uucp/Sysfiles** file to specify other files to use for Dialers configuration. Refer to Sysfiles File Format for BNU in *AIX 5L Version 5.2 Files Reference* for more information.

7. Decide how much access to your system you want to provide to each remote system you call and to each remote system that calls you. Set up appropriate entries for each system and each login name in the **Permissions** file. Refer to Permissions File Format for BNU in *AIX 5L Version 5.2 Files Reference* for details.

8. Issue the **uucheck** command to verify that everything is in place:

   ```
   /usr/sbin/uucp/uucheck –v
   ```

   The **uucheck** command verifies that the directories, programs, and support files are set up properly and that the **Permissions** file entries are consistent. If the **uucheck** command reports any errors, fix the errors.

9. If you wish, set up automatic monitoring of BNU operations and automatic polling of remote systems (see Setting Up Automatic Monitoring of BNU and Setting Up BNU Polling of Remote Systems).

Setting Up Automatic Monitoring of BNU

Prerequisites

- Complete the steps listed in Configuring BNU.
- You must have root user authority to edit the **/var/spool/cron/crontabs/uucp** file.

Procedure

BNU uses the **cron** daemon to start BNU daemons and to monitor BNU activity. The **cron** daemon reads the **/var/spool/cron/crontabs/uucp** file for instructions about when to start BNU procedures.

1. Log in as a user with root user authority.
2. Using an ASCII text editor, edit the **/var/spool/cron/crontabs/uucp** file.
3. Uncomment the lines for the BNU maintenance procedures, **uudemon.admin** and **uudemon.cleanup**. You can change the times these procedures are run if your system needs maintenance at more or less frequent intervals. It is best, however, to run the **uudemon.admin** command at least once a day and the **uudemon.cleanup** command at least once a week.
4. You can use the **crontabs/uucp** file to schedule other BNU maintenance commands, such as the **uulog**, **uuclean**, or **uucleanup** commands. In addition, you can use the **crontabs/uucp** file to instruct the **cron** daemon to start the **uucico**, **uuxqt**, or **uusched** daemons at specific times.

Setting Up BNU Polling of Remote Systems

Prerequisites

1. Complete the steps listed in Configuring BNU on page 9-10.
2. You must have root authority to edit the **/var/spool/cron/crontabs/uucp** file and the **/etc/uucp/Poll** file.

Procedure

To enable BNU to poll remote systems for jobs, list the systems in the **/etc/uucp/Poll** file. In addition, run the **uudemon.hour** and **uudemon.poll** commands periodically.

1. Decide which remote systems to automatically poll. Decide how often you want to poll each one. Specify times for each system with the **Poll** file as seldom as once a day or as often as you wish.
2. Log in as a user with root authority.
3. Using an ASCII text editor or the **uucpadm** command, edit the **Poll** file. Add an entry for each system your system will poll.

   Note:: The systems listed in the **Poll** file must also be listed in the **/etc/uucp/Systems** file.
4. Using an ASCII text editor, edit the **/var/spool/cron/crontabs/uucp** file. Remove the comment characters (#) from the lines that run the **uudemon.hour** and **uudemon.poll** commands. You can change the times these commands are run. However, be sure to schedule the **uudemon.poll** command approximately five minutes *before* you schedule the **uudemon.hour** command.

BNU will now automatically poll the systems listed in the **Poll** file at the times you have specified.

Using the /etc/uucp/Systems File

The remote systems are listed in the **/etc/uucp/Systems** files. The **/etc/uucp/Systems** file is the default **Systems** file. The system administrator can specify additional files in the **/etc/uucp/Sysfiles** file.

Each entry in a **Systems** file contains:

- Name of the remote system
- Times when users can connect to the remote system
- Type of link (direct line or modem)
- Speed of transmission over the link
- Information needed to log in to the remote system.

Each entry in a **Systems** file represents one remote system. To establish communications, the remote system must be listed in the local **Systems** file. A **Systems** file must be present on every system that uses the BNU facility. Normally, only the root user can read the **Systems** files. Any user, however, can list the names of remote BNU systems using the **uuname** command.

Editing Devices Files for Hardwired Connections

Prerequisites

You must have root authority to edit the **/etc/uucp/Devices** file or another file specified in **/etc/uucp/Sysfiles** as a **Devices** file.

Procedure to Set Up a System Name Entry

To set up a hardwired connection specifying a port and a remote system, make an entry as follows:

1. Enter the name of the remote system to which you want to connect the local computer over the hardwired line in the *Type* field in the second line of the entry.
2. Enter the device name appropriate for the hardwired connection used at your site in the *Line* field in both lines of the entry.
3. Enter a – (hyphen) for a placeholder in the *Line2* field in both lines of the entry.
4. Enter the transmission rate appropriate for the hardwired connection used at your site in the *Speed* field in both lines of the entry.
5. Enter `direct` (all lowercase) in the *Dialer–Token Pairs* field in both lines of the entry.

For example:

```
type device – speed direct
```

Continue adding entries to the **Devices** file until you have listed each hardwired device connecting the local system to a remote system.

Procedure to Set Up a Direct Entry

To set up a hardwired connection between two systems that use a permanent asynchronous serial connection, make a one–line entry as follows:

1. Enter the name of the remote system in the first (*Type*) field.
2. Enter the name of the tty device in the second (*Line*) field.
3. Enter a – (hyphen) for a placeholder in the third (*Line2*) field.
4. Enter the transmission rate appropriate for the hardwired connection used at your site in the fourth (*Class*) field.
5. Enter `direct` (all lowercase) in the fifth (*Dialer–Token Pairs*) field.

For example:

```
type device – speed direct
```

Continue adding entries to the **Devices** file until you have listed each hardwired device connecting the local system to a remote system.

Editing Devices File for Autodialer Connection

Prerequisites

You must have root authority to edit the **/etc/uucp/Devices** file or another file specified in **/etc/uucp/Sysfiles** as a **Devices** file.

Procedure

In telephone–connection entries, the *Type* field is specified as an automatic calling unit (ACU). Type ACU as the *Type* field entry in all remote connections established over a phone line. To set up **Device** file entries for autodialer connections, make a one–line entry for each modem:

1. Enter `ACU` in the first (*Type*) field.
2. The second (*Line*) field contains the name of the device that is attached to the modem. Enter the device name appropriate for your site.
3. Enter a `–` (hyphen) as a placeholder in the third (*Line2*) field, unless the autodialer is a standard 801 dialer. If the autodialer is a standard 801 dialer, enter `801`.
4. In the fourth (*Speed*) field, enter the baud rate appropriate for your modem and line (this can be 300, 1200, 2400, or higher, depending on the modem) or the class of your modem (for example, D2400).

   Note:: If the modem can be used at more than one specific rate, make a separate entry in the **Devices** file for each rate. If the modem can be used at any rate, enter the word `Any` in the *Speed* field.
5. Enter the name of the modem as the *Dialer* field entry in the fifth (*Dialer–Token Pair*) field. If you are planning to include complete phone numbers in the **/etc/uucp/Systems** file or another **Systems** file specifies in **/etc/uucpSysfiles**, leave the *Token* field blank. (A blank instructs the BNU program to use the default `\D` token.) If you are planning to use dialing–code abbreviations specified in the **/etc/uucp/Dialcodes** file, enter the token `\T`.

For example:

```
type line – speed dialer – token pair
```

Continue adding entries to the **Devices** file until you have listed each connection between the local system and a remote system that uses a telephone line and a modem.

Editing Devices File for TCP/IP

Prerequisites

You must have root authority to edit the **/etc/uucp/Devices** file or another file specified in **/etc/uucp/Sysfiles** as a **Devices** file.

Procedure

If your site is using the TCP/IP system, include the relevant TCP/IP entry in the **Devices** file. To set up the file for use with the TCP/IP system, enter the following line in the **Devices** file:

```
TCP – – – TCP
```

Maintaining BNU

BNU must be maintained to work correctly on your system. To maintain BNU:

- Read and remove log files periodically.
- Use the **uuq** and **uustat** commands to check the BNU queues to ensure jobs are transferring to remote systems correctly.
- Schedule automatic commands that poll remote systems for jobs, return unsent files to users, and send you periodic messages about BNU status.
- Periodically update the configuration files to reflect changes in your system.

In addition, occasionally check with administrators of remote systems to keep up with changes on their systems that might affect your configuration. For example, if the supervisor of system `venus` changes your system password, you must put the new password in the **/etc/uucp/Systems** file (or the appropriate Systems file specified by **/etc/uucp/Sysfiles**) before your system can log in to system `venus`.

For a list of commands used to maintain BNU, see BNU Files, Commands, and Directories Reference on page 9-38.

Working with BNU Log Files

BNU creates log files and error files to track its own activities. These files must be checked and removed periodically to keep them from filling the storage space on your system. BNU provides several commands for use in cleaning log files:

- **uulog**
- **uuclean**
- **uucleanup**
- **uudemon.cleanu**.

Run these commands manually or use entries in the **/var/spool/cron/crontabs/uucp** file to run the commands by the **cron** daemon.

Log Files in the .Log and .Old Directories

BNU creates individual log files in the **/var/spool/uucp/.Log** directory. BNU creates these log files for each accessible remote system, using the **uucp**, **uucico**, **uux**, and **uuxqt** commands. BNU places status information about each transaction in the appropriate log file each time someone on the system uses BNU. When more than one BNU process is running the system cannot access the log file. Instead, it places the status information in a separate file with a**.LOG** prefix.

The **uulog** command displays a summary of **uucp** or **uux** requests, by user or by system. The **uulog** command displays the files. However, you can also have BNU automatically combine the log files into a primary log file. This is called *compacting* the log files and can be done with the **uudemon.cleanu** command, usually run by the **cron** daemon.

The **cron** daemon runs the **uudemon.cleanu** command. The **uudemon.cleanu** command combines the **uucico** and **uuxqt** log files on the local system and stores them in the **/var/spool/uucp/.Old** directory. At the same time, the command removes old log files previously stored in the**.Old** directory. By default, the **uudemon.cleanu** command saves log files that are two days old.

If storage space is a problem, consider reducing the number of days that files are kept. To track BNU transactions over a longer period of time, consider increasing the number of days that files are kept. To change the default time for saving log files, modify the shell procedure for the **uudemon.cleanu** command. This script is stored in the **/usr/sbin/uucp** directory and can be modified with root authority.

Other BNU Log Files

BNU also collects information and stores it in the **/var/spool/uucp/.Admin** directory. This directory contains the **errors**, **xferstats**, **Foreign**, and **audit** files. These files must be checked and removed occasionally to save storage space. BNU creates each file when it is needed.

When another system contacts your system with the **uucico** daemon debugging mode on, it invokes the **uucico** daemon on your system with debugging turned on. The debugging messages generated by the daemon on the local system are stored in the **audit** file. This file can get quite large. Check and remove the **audit** file often.

The **errors** file records errors encountered by the **uucico** daemon. Checking this file can help you correct problems such as incorrect permissions on BNU work files.

The **xferstats** file contains information about the status of every file transfer. Check and remove this file occasionally.

The **Foreign** file is important to the security of your system. Whenever an unknown system attempts to log in to the local system, BNU calls the **remote.unknown** shell procedure. This shell procedure logs the attempt in the **Foreign** file. The **Foreign** file contains the names of the systems that have attempted to call the local system and been refused. If a system has been attempting frequent calls, use this information when considering whether to allow that system access.

Systemwide Log Files used by BNU

Because many BNU processes need root authority to complete their tasks, BNU creates frequent entries in the **/var/spool/sulog** log file. Similarly, using the **cron** daemon to schedule BNU tasks creates multiple entries in the **/var/spool/cron/log** file. When using BNU, check and clean these files.

BNU Maintenance Commands

The Basic Networking Utilities contain several commands for monitoring BNU activities and cleaning BNU directories and files.

Cleanup Commands

BNU contains three commands that clean directories and remove files that have not been sent:

uuclean Deletes all files older than a specified number of hours, from the BNU administrative directories. Use the **uuclean** command to specify a directory to be cleaned or a type of file to be deleted. You can also instruct the command to notify the owners of the deleted files. The **uuclean** command is the Berkeley equivalent of the **uucleanup** command.

uucleanup Performs functions similar to the **uuclean** command. However, the **uucleanup** command checks the age of files based on *days* rather than hours. Use the **uucleanup** command to send a warning message to users whose files have not been transferred, notifying them that the files are still in the queue. The **uucleanup** command also removes files relating to a specified remote system.

uudemon.cleanu A shell procedure that issues the **uulog** and **uucleanup** commands to compress the BNU log files and remove log and work files over three days old. The **uudemon.cleanu** command is run by the **cron** daemon.

Status–checking Commands

BNU also provides commands for checking the status of transfers and log files:

uuq
Displays jobs currently in the BNU job queue. Use the **uuq** command to display the status of a specified job or of all jobs. With root authority, you can use the **uuq** command to delete a job from the queue.

uustat
Provides information similar to that provided by the **uuq** command, in a different format. Use the **uustat** command to check the status of jobs and delete jobs you own. With root authority, you can also delete jobs belonging to other users.

uulog
Displays a summary of **uucp** or **uux** requests, by user or by system. The **uulog** command displays the file names. See Working with BNU Log Files.

uupoll
Forces a poll of a remote system. This is helpful when work for that system is waiting in the queue and needs to be transferred, before the system is scheduled to be called automatically.

uusnap
Displays a very brief summary of BNU status. For each remote system, this command shows the number of files awaiting transfer. However, it does not show how long they have been waiting. The **uusnap** command is the Berkeley equivalent of the **uustat** command.

Shell Procedures

BNU is delivered with two shell procedures used for maintenance:

uudemon.cleanu
Discussed in Cleanup Commands on page 9-18.

uudemon.admin
Issues the **uustat** command The **uustat** command reports the status of BNU jobs. It sends the results to the uucp login ID as mail. You can modify the **uudemon.admin** shell procedure to send the mail elsewhere, or use a mail program to reroute all mail for the uucp login ID to the user responsible for BNU administration.

These shell procedures are stored in the **/usr/sbin/uucp** directory. Copy the procedures and modify the copy, if you want to change what they do. Run the procedures from the command line or schedule them to be run by the **cron** daemon.

To automatically run the **uudemon.cleanu** and **uudemon.admin** commands, remove the comment characters (#) from the beginning of the relevant lines in the **/var/spool/cron/crontabs/uucp** file.

Monitoring a BNU Remote Connection

Prerequisites

- The BNU program must be installed on your system.
- A link (hardwired, modem, or TCP/IP) must be set up between your system and the remote system.
- The BNU configuration files, including the **Systems** file, **Permissions** file, **Devices** file, and **Dialers** file (and **Sysfiles** file, if applicable), must be set up for communications between your system and the remote system.

  Note:: You must have root user authority to modify the BNU configuration files.

Procedure

The **Uutry** command can help you monitor the **uucico** daemon process if users at your site report file–transfer problems.

1. Issue the **uustat** command to determine the status of all the transfer jobs in the current queue as follows:

   ```
   uustat –q
   ```

   The system displays a status report like the following:

   ```
   venus 3C (2) 05/09–11:02 CAN'T ACCESS DEVICE
   hera 1C 05/09–11:12 SUCCESSFUL
   merlin 2C 5/09–10:54 NO DEVICES AVAILABLE
   ```

   This report indicates that three command (**C**.*) files intended for remote system `venus` have been in the queue for two days. There could be several reasons for this delay. For example, perhaps system `venus` has been shut down for maintenance or the modem has been turned off.

2. Before you begin more extensive troubleshooting activities, issue the **Uutry** command as follows to determine whether your local system can contact system `venus` now:

   ```
   /usr/sbin/uucp/Uutry –r venus
   ```

   This command starts the **uucico** daemon with a moderate amount of debugging and the instruction to override the default retry time. The **Uutry** command directs the debugging output to a temporary file, `/tmp/venus`.

3. If your local system succeeds in establishing a connection to system `venus`, the debugging output contains a good deal of information. However, the final line in this script, which follows, is the most important:

   ```
   Conversation Complete: Status SUCCEEDED
   ```

   If the connection is successful, assume that the temporary file–transfer problems are now resolved. Issue the **uustat** command again to make certain that the files in the spooling directory have been transferred successfully to the remote system. If they have not, use the steps in Monitoring a BNU File Transfer to check for file–transfer problems between your system and the remote system.

4. If your local system cannot contact the remote system, the debugging output generated by the **Uutry** command contains the following type of information (the exact form of the output might vary):

   ```
   mchFind called (venus)
   conn (venus)
   getto ret –1
   Call Failed: CAN'T ACCESS DEVICE
   exit code 101
   Conversation Complete: Status FAILED
   ```

   First, check the physical connections between the local and remote systems. Make sure that the remote computer is turned on and all cables are properly connected, that the ports are enabled or disabled (as appropriate) on both systems, and that the modems (if applicable) are working.

   If the physical connections are correct and secure, then verify all the relevant configuration files on both the local and remote systems, including the following:

   – Make certain that the entries in the **Devices**, **Systems**, and **Permissions** files (and **Sysfiles** file, if applicable) in the **/etc/uucp** directory are correct on both systems.

   – If you are using a modem, make sure that the **/etc/uucp/Dialers** file (or an alternate file specified in **/etc/uucp/Sysfiles**) contains the correct entry. If you are using dial–code abbreviations, be sure the abbreviations are defined in the **/etc/uucp/Dialcodes** file.

   – If you are using a TCP/IP connection, make sure that the **uucpd** daemon can be run on the remote system and that the configuration files contain the correct TCP entries.

5. Once you have checked the physical connections and configuration files, issue the **Uutry** command again. If the debugging output still reports that the connection failed, you might need to confer with a member of your systems support team. Save the debugging output produced by the **Uutry** command. This might prove helpful in diagnosing the problem.

Monitoring a BNU File Transfer

Prerequisites

1. The BNU program must be installed on and configured for your system
2. Establish a connection to a remote system using the steps given in Monitoring a BNU Remote Connection.

Monitoring a File Transfer

Use this procedure to monitor a file transfer to a remote system. Monitoring a file transfer is useful when file transfers to the remote system in question are failing for unknown reasons. The debugging information produced by the **uucico** daemon (called by the **Uutry** command) can help you find out what is working incorrectly.

The **Uutry** command enables you to monitor file transfers, as follows:

1. Prepare a file for transfer using the **uucp** command with the **–r** flag by entering:

```
uucp –r test1 venus!~/test2
```

   The **–r** flag instructs the UUCP program to create and queue all necessary transfer files but *not* to start the **uucico** daemon.

2. Issue the **Uutry** command with the **–r** flag to start the **uucico** daemon with debugging turned on by entering:

```
/usr/sbin/uucp/Uutry –r venus
```

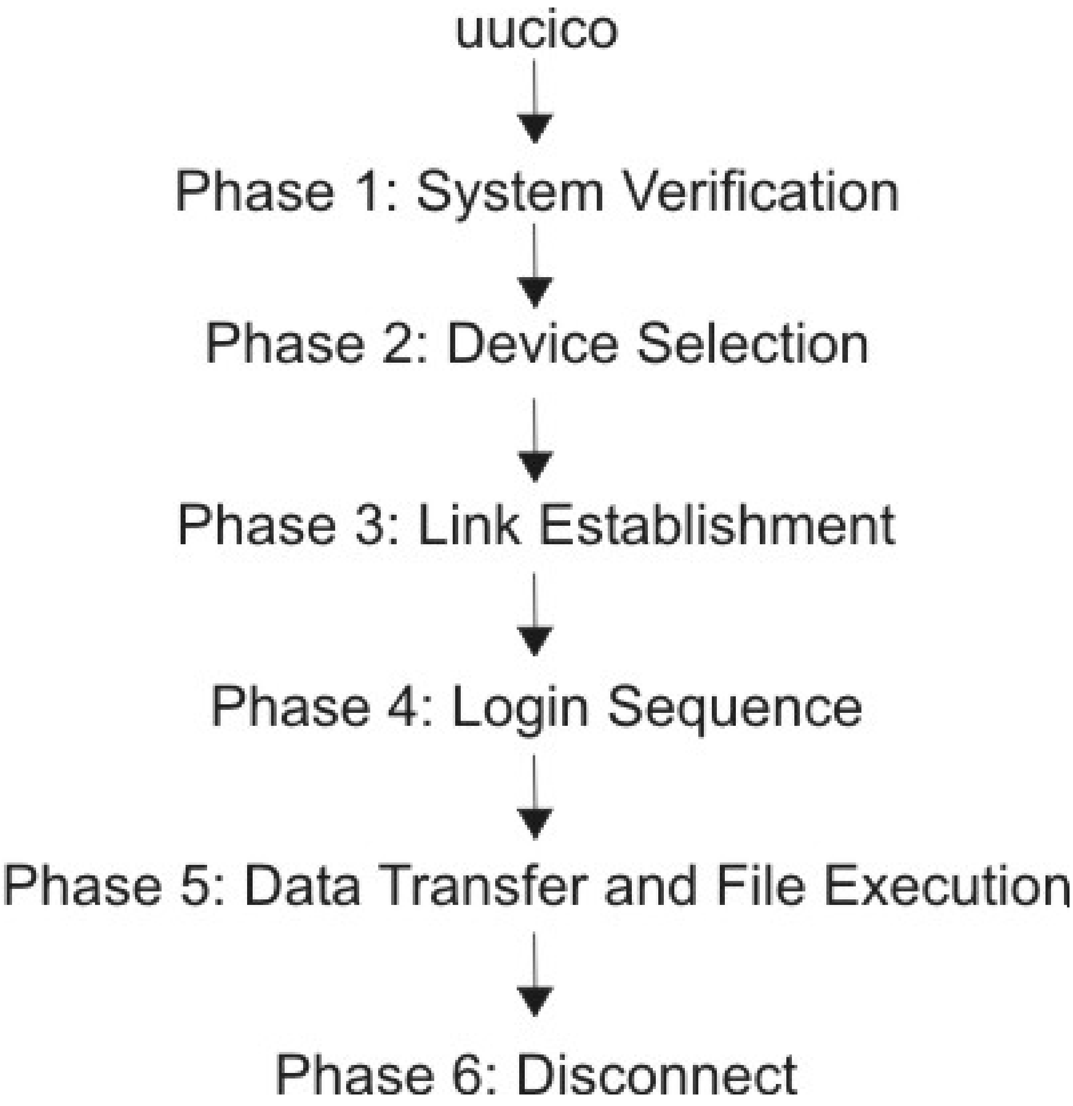
   This instructs the **uucico** daemon to contact remote system `venus` overriding the default retry time. The daemon contacts system `venus`, logs in, and transfers the file, while the **Uutry** command produces debugging output that enables you to monitor the **uucico** process. Press the Interrupt key sequence to stop the debugging output and return to the command prompt.

   The **Uutry** command also stores the debugging output in the **/tmp/** *SystemName* file. If you break out of the debugging output before the connection is complete, you can page through the output file to see the outcome of the connection.

Debugging BNU Problems

BNU error messages can be linked to a specific phase in the conversation flow. Use the ”BNU Conversation Flow Diagram” and the following error descriptions to help diagnose your BNU problems. Some of the following messages might not be sent from BNU, but are included in case another UUCP version is in use.

Figure 36. BNU Conversion Flow Diagram This illustration shows the flow and different phases of BNU conversion. From uucico at the top, data is passed to Phase 1–System Verification, then Phase 2–Device Selection, and Phase 3–Link Establishment, then Phase 4–Login Sequence, next Phase 5–Data Transfer and File Execution and last, Phase 6–Disconnect.


uucico
Phase 1: System Verification
Phase 2: Device Selection
Phase 3: Link Establishment
Phase 4: Login Sequence
Phase 5: Data Transfer and File Execution
Phase 6: Disconnect

PHASE 1 Status Messages

`Assert Error`

The local system unit is having problems. Check the error report for possible causes by issuing the command **`errpt –a | pg`**.

`System not in Systems`

If you supply a remote system name that is not found in the **Systems** files, this status message is created, BNU will terminate. Use the **uuname** command to check the system name again.

`Wrong time to call`

The **Systems** file has restrictions on times to allow outgoing calls. BNU will keep trying until the time is right. Check the **Systems** file.

`Callback required`

The network has restricted usage either for security or economic reasons, and access is denied at this time.

`Cannot call`
`No Call`

These errors mean BNU recently tried to call the remote system and failed. It will not immediately try again. They can also be caused by an old system status file being retained thus keeping the **uucico** daemon from trying again.

PHASE 2 Status Messages

`Dialer Script Failed`

Your **Dialers** file script did not complete successfully.

`No Device Available`
`Can’t Access Device`

The modem or the outgoing phone line from your system is busy. Check for an error in the device entry of the **Systems** file. Also, check the **Devices** and **Dialers** files to be sure logical devices have physical devices associated with them. The file **/etc/uucp/Sysfiles** might be specifying an alternate **Systems**, **Devices**, or **Dialers** file that is not correctly configured. Is the device in use by some other program? Check the **/var/locks** directory for lock on port. If a lock file exists (for example, **LCK..TTY0**), check to see if the process identified by the number in the lock file is still active. If not, you can remove it (for example, `rm /var/locks/LCK..TTY0`). Also check the permissions on the port.

`Dial Failed`
`Failed (call to system)`

These errors appear when your system dials another successfully but the other system does not answer. It might also indicate a problem in the **Devices** files. Enter the command `uucico –r1 –x6 –s SystemName`. It could be that BNU is expecting some string that it is not receiving. Make the connection by hand to find out what needs to be incorporated into the **Systems** files entry to satisfy the request. Please keep ”timing” in mind; perhaps some delays in the modem dial string are needed. This could also mean that the port is busy, you dialed an incorrect number, or BNU lost ownership of the port.

`OK`
`Auto Dial`

These are informative messages only and do not indicate an error.

PHASE 3 Status Messages

`Handshake Failed (LCK)`
The device is being used by someone else; the process could not create the **LCK** file. Sometimes **LCK** files must be manually removed by the administrator. After a number of retries, see your system administrator. See if another process has control of the port (for example, another instance of the **uucico** daemon).

`Login Failed`
The login failed due to a bad connection or possibly a slow machine.

`Timeout`
The remote system did not respond within a set period of time. This could also indicate a problem with the chat script.

`Succeeded (Call to System)`
The call was completed.

`BNU (continued)`
These are informative messages only and do not indicate an error.

PHASE 4 Status Messages

`Startup Failed`
`Remote reject after login`
After login, the **uucico** daemon is started on the remote system. If there is a problem initiating a conversation between the two systems, these messages are created. You might have also logged into the incorrect BNU account or the initial handshake failed.

`Wrong machine name`
A machine was called incorrectly or the machine name was changed.

`Bad login/machine combination`
The login to the remote system failed. The problem could be an incorrect phone number, an incorrect login or password, or an error in the chat script.

`Remote has a LCK file for me`
Both systems were simultaneously trying to call each other. The local request will fail temporarily.

`OK`
`Talking`

These are informative messages only and do not indicate an error.

`LOGIN:`
`PASSWORD:`

If the login or password prompt is in all capital letters, the modem might be in echo mode (E1 on Hayes compatibles). This causes the modem to echo back, or send, a RING to your system when an incoming call is received. The **getty** command receives the string and accordingly changes the `login:` or `password:` into all caps. Change the echo mode on the modem to off (use `ATE0` for Hayes compatibles).

Note::

Keep in mind that once this change is made, you should use `ATE1` in the chat script of your **Dialers** files, or you will not get the expected `OK` back from the modem.

If the remote port is set for `delay` or `getty –r` and the chat script expects key input, then the ports set for `delay` are expecting one or more carriage returns before proceeding with the login. Try beginning the chat script on the dialing system with the following:

```
”” \r\d\r\d\r\d\r in:––in:  ...
```

Interpreted, this chat script reads as follows: expect nothing, send return, delay, return, delay, return, delay, return.

PHASE 5 Status Messages

`Alarm`

The **uucico** daemon is having trouble with the connection. Either the connection is bad or ”xon/xoff” is set to yes on the modem.

`Remote access to path/file denied`
`copy (failed)`

These messages indicate a permission problem; check file and path permissions.

`Bad read`

The remote system ran out of space, most likely in the spool area, or the **uucico** daemon could not read or write to device.

`Conversation failed`

The modem carrier detect was lost. Possibly the modem was turned off, the cable is loose or disconnected, or the remote system crashed or is shut down. Telephone disconnection can also cause this error.

`Requested`
`Copy (succeeded)`

These are informative messages only and do not indicate an error.

PHASE 6 Status Messages

`OK`
`(Conversation Complete)`
	The remote system can deny the hangup request and reverse the roles (meaning the remote system has work for the local system to do). Once the two **uucico** daemons agree that no more work exists, they hang up.

`Conversation succeeded`	This is an informative message only and does not indicate an error.

Debugging BNU Login Failures Using the uucico Daemon

Prerequisites

- BNU must be installed on your system.
- A link (hardwired, modem, or TCP/IP) must be set up between your system and the remote system.
- The BNU configuration files, including the **Sysfiles** file (if applicable), the **Systems** file, **Permissions** file, **Devices** file, and **Dialers** file, must be set up for communications between your system and the remote system.

  Note:: You must have root user authority to modify the BNU configuration files.
- You must have root user authority to invoke the **uucico** daemon in debugging mode.

Procedure

1. To produce debugging information about a local–to–remote system connection that is not working, start the **uucico** daemon with the **–x** flag as follows:

   ```
   /usr/sbin/uucp/uucico –r 1 –s venus –x 9
   ```

   where `–r 1` specifies the master, or caller mode; `–s venus`, the name of the remote system to which you are trying to connect; and `–x 9`, the debug level that produces the most detailed debugging information.
2. If the expect–send sequence entry in a **Systems** file in the format of **/etc/uucp/Systems** is:

   ```
   venus Any venus 1200 – ”” \n in:––in: uucp1 word:
     mirror
   ```

   the **uucico** daemon connects the local system to the remote system `venus.` The debugging output is similar to:

   ```
   expect: ””
   got it
   sendthem (^J^M)
   expect (in:)^
   M^Jlogin:got it
   sendthem (uucp1^M)
   expect (word:)^
   M^JPassword:got it
   sendthem (mirror^M)
   imsg >^M^J^PShere^@Login Successful: System=venus
   ```

   where:

`expect: ””`	Specifies that the local system will not wait for any information from the remote system.
`got it`	Acknowledges that the message has been received

`sendthem (^J^M)`
Specifies that the local system will send the remote system a carriage return and a new line.

`expect (in:)`
Specifies that the local system expects to receive the remote system login prompt, which ends in the character string.

`^M^Jlogin:got it`
Confirms that the local system received the remote login prompt.

`sendthem (uucp1^M)`
Specifies that the local system will send the `uucp1` login ID to the remote system.

`expect (word:)`
Specifies that the local system expects to receive the remote system password prompt, which ends in the `word:` character string.

`^M^JPassword:got it`
Confirms the local system received the remote password prompt.

`sendthem (mirror^M)`
Specifies that the local system will send the password for the `uucp1` login ID to the remote system.

```
imsg >^M^J^PShere^@Login
Successful: System=venus
```
Confirms the local system is successfully logged in to remote system `venus`.

Notes:

1. The expect–send debugging output produced by the **uucico** command can come either from information in the **/etc/uucp/Dialers** file or from information in the **/etc/uucp/Systems** file. Information about communication with the modem comes from the **Dialers** file, while information about communication with the remote system comes from the **Systems** file. (Note that **/etc/uucp/Systems** and **/etc/uucp/Dialers** are default BNU configuration files. Other files can be specified in **/etc/uucp/Sysfiles** to serve the same role.)
2. To set up a connection with a remote system, you must be familiar with the login sequence of that system.

Contacting Connected UNIX Systems Using the tip Command

Use the **tip** command to contact any connected system running the UNIX operating system. The **tip** command is installed with the Basic Networking Utilities (BNU) and can use the same asynchronous connections used by BNU.

The **tip** command uses variables and escape signals, as well as flags, to control its operations. The flags can be entered at the command line. The escape signals can be used over a connection with a remote system to start and stop file transfers, change the direction of a file transfer, and exit to a subshell.

tip Command Variables

The **tip** command variables define settings such as the end–of–line character, the break signal, and the mode of file transfers. Variable settings can be initialized at run time using a**.tiprc** file. Variable settings can also be changed during execution using the **~s** escape signal. Some variables, such as the end–of–line character, can be set for an individual system in the system entry ofthe **remote** file.

The **tip** command reads three files, the **phones** file, **remote** file, and**.tiprc** file, to determine initial settings for its variables. The**.tiprc** file must always be in the user home directory. The names and locations of the **remote** and **phones** files can vary. The names of the **remote** file and the **phones** file can be determined by environment variables:

PHONES Specifies the name of the user phone file. The file can have any valid file name and must be set up in the format of the file **/usr/lib/phones–file**. The default file is **etc/phones**. If a file is specified with the **PHONES** variable, it is used in place of (not in addition to) the **/etc/phones** file.

REMOTE Specifies the name of the user remote system definition file. The file can have any valid file name and must be set up in the format of the **/usr/lib/remote–file** file. The default file is **/etc/remote**. If a file is specified with the **REMOTE** variable, it is used in place of (not in addition to) the **/etc/remote** file.

To use an environment variable, set it before starting the **tip** command. As an alternative, the names of the **phones** and **remote** files can be determined using the **tip** command **phones** variable and **remote** variable, respectively, in the**.tiprc** file.

Note:: The **tip** command reads only the *last* **remote** or **phones** file specified. Thus, if you specify a **remote** or **phones** file with a variable, the new file is used in place of (not in addition to) any previous files you specified.

The **tip** command uses variable settings in the following order:

1. The command checks the settings of the **PHONES** and **REMOTE** environment variables for the files to use as the **phones** and **remote** files.
2. The command reads the**.tiprc** file and sets all variables accordingly. If the **phones** or **remote** variable is set in the**.tiprc** file, this setting overrides the environment variable setting.
3. When a connection to a remote system is initiated, the command reads the **remote** file entry for that system. The settings in the **remote** file entry override settings made in the**.tiprc** file.
4. If the **– BaudRate** flag is used with the **tip** command, the specified rate overrides all previous baud rate settings.
5. A setting made with the **~s** escape signal overrides all previous settings of a variable.

   Note:: Any **tip** user can create a**.tiprc** file and use this file to specify initial settings for **tip** variables. The**.tiprc** file must be placed in the user **$HOME** directory.

tip Command Configuration Files

Before the **tip** command can connect to a remote system, the **/etc/remote** and **/etc/phones** files must be established.

/etc/remote	Defines attributes of remote systems such as the port and type of device to use to reach the system, as well as the signals to use to indicate the beginnings and endings of transmissions.
/etc/phones	Lists telephone numbers used to contact remote systems over a modem line.

To establish one of these files, copy a sample file to the correct name and modify it to suit the needs of your site. Sample **remote** and **phones** files are delivered with the bos.net.uucp package. The sample **remote** file is named **/usr/lib/remote–file.** The sample **phones** file is named **/usr/lib/phones–file**. Copy **/usr/lib/remote–file** to **/etc/remote** and modify **/etc/remote**.

A **tip** user can also create customized **remote** and **phones** files. An individual **remote** file must be in the format of the **/usr/lib/remote–file** file and specified with the **remote** variable or the **REMOTE** environment variable. An individual **phones** file must be in the format of the **/usr/lib/phones–file** file and specified with the **phones** variable or the **PHONES** environment variable. If an individual **phones** or **remote** file is specified with one of the variables, that file is read in place of (not in addition to) the **/etc/phones** or **/etc/remote** file.

Users of **tip** can use combinations of individual **phones** and **remote** files. For example, a user could use the default **remote** file, **/etc/remote**, but use an individual **phones** file named with the **phones** variable**.**

BNU Configuration Files

Basic Network Utilities (BNU) uses the following configuration files:

/etc/uucp Contains all the configuration files for BNU.

/var/spool/uucppublic Contains files that have been transferred.

/etc/uucp/Systems Contains a list of systems to which the **uucico** program can connect.

/etc/uucp/Devices Defines the device type, location, speed, and other basic communication parameters for many system dial–out programs. Only dial–out connections use this file.

/etc/uucp/Permissions Creates security control, with limitations, over machines attempting to communication with your machine.

/etc/uucp/Dialers Specifies the dialer types. Each dialer uses a specific command set when attempting to dial the modem. The most common dialer types are `hayes`, `direct`, and `TCP` (Transmission Control Protocol).

/etc/uucp/Dialcodes Makes standardized names for certain parts of a phone number. For example, if you frequently make calls to a certain area code in San Francisco, you could create the following entry: `SF09,1415`.

/etc/uucp/Sysfiles Enables a BNU administrator to specify files to fill the role of BNU configuration files other than **/etc/uucp/Systems**, **/etc/uucp/Devices**, and **/etc/uucp/Dialers**. Distinctions can be made between what files are used for **uucico** traffic versus **cu** –related (**cu**, **ct**, **slattach**) activity.

/usr/sbin/uucp/remote.unknown Defines a shell script. It is run by the BNU program when a remote computer that is not listed in the local permissions file attempts to communicate with that local system.

/etc/uucp/Poll Schedules polling of passing systems. Its format is similar to the **crontab** file. Poll format is *SiteName*, a tab, and the hours to poll (0–23), separated by spaces.

Correlation of Files

Systems file:	`SystemName Any v32ibm 9600 555–1111`
Devices file:	`v32ibm tty0 – Any ibm \D`
Dialers file:	`ibm =, –, #" \d ATSFI\r\c#OK#AFE1SD3L2MIC0SCI\r\c#OK...`

BNU Configuration for a TCP/IP Connection Example

The following files are set up for a Transmission Control Protocol/Internet Protocol (TCP/IP) connection between systems `zeus` and `hera`, where `zeus` is considered the local system and `hera` the remote system.

Entries in the Local System Files

Files entries on local system `zeus` include the following:

Systems File

The **Systems** file on system `zeus` should contain the following entry to allow `zeus` to contact system `hera`:

```
hera Any TCP,t – – in:––in: uzeus word: birthday
```

This specifies that system `zeus` can call system `hera` at any time, using the **t** protocol for communications with system `hera`. System `zeus` logs in to system `hera` as `uzeus` with the password `birthday`.

Note:: The **t** protocol supports the **tcp** protocol. Therefore, always use the **t** protocol for BNU communications over TCP/IP connections. However, the **t** protocol cannot be used when the *Type* field is `ACU` (automatic calling unit) or when a modem connection is being used.

BNU uses the *Type* and *Class* fields in the **Systems** file to find the appropriate device for the connection. Accordingly, it checks the **Devices** file for an entry of type `TCP`.

Devices File

A **Devices** file used by the **uucico** daemon on system `zeus` should contain the following entry for TCP/IP connections:

```
TCP  –  –  –  TCP
```

Because the device type is `TCP`, there are no *Class*, *Line*, or *Line2* entries. The *Dialer* is also specified as `TCP`. Accordingly, BNU looks in the **Dialers** files for a `TCP` entry.

Dialers File

The **Dialers** file used by the **uucico** daemon on system `zeus` should contain a TCP/IP entry as follows:

```
TCP
```

This entry specifies that no dialer configuration is required.

Note:: Dialer configuration is never required over a TCP/IP connection.

Permissions File

The **Permissions** file on system `zeus` contains the following entry granting system `hera` access to system `zeus`:

```
LOGNAME=uhera SENDFILES=yes REQUEST=yes \
MACHINE=zeus:hera VALIDATE=uhera /
READ=/var/spool/uucppublic:/home/hera \
WRITE=/var/spool/uucppublic:/home/hera COMMANDS=ALL
```

This combined LOGNAME and MACHINE entry provides the following permissions to system hera on system zeus:

- System hera can request and send files regardless of who initiated the call.
- System hera can read and write to the public directory and the **/home/hera** directory on system zeus.
- System hera can execute all commands on system zeus.
- System hera must log in to system zeus as user uhera and cannot use any other login ID for BNU transactions.

  Note:: Because the permissions are the same regardless of which system initiates the call, the preceding LOGNAME and MACHINE entries are combined. Separately, they are:

```
LOGNAME=uhera VALIDATE=hera SENDFILES=yes REQUEST=yes& \
READ=/var/spool/uucppublic:/home/hera \
WRITE=/var/spool/uucppublic:/home/hera

MACHINE=zeus:hera REQUEST=yes COMMANDS=ALL\
READ=/var/spool/uucppublic:/home/hera \
WRITE=/var/spool/uucppublic:/home/hera
```

Entries in the Remote System's Files

Files on remote system hera include the following:

Systems File

A **Systems** file on system hera should contain the following entry to allow hera to contact system zeus:

```
zeus Any TCP,t – – ogin:––ogin: uhera ord: lightning
```

This specifies that system hera can call system zeus at any time, using the **t** protocol for communications with system zeus. System hera logs in to system zeus as user uhera with the password lightning. Again, BNU next checks the **Devices** files for an entry of type TCP.

Note:: The **t** protocol supports the **tcp** protocol. Therefore, always use the **t** protocol for BNU communications over TCP/IP connections. However, the **t** protocol cannot be used when the *Type* field is ACU or when a modem connection is being used.

Devices File

The **Devices** file used by the **uucico** daemon on system hera should contain the following entry for TCP/IP connections:

```
TCP  –  –  –  TCP
```

Because the device type is TCP, there are no *Class*, *Line*, or *Line2* entries. The *Dialer* is also specified as TCP. Accordingly, BNU looks in the **Dialers** files for a TCP entry.

Dialers File

The **Dialers** file used by the **uucico** daemon on system hera should contain a TCP/IP entry as follows:

```
TCP
```

This entry specifies that no dialer configuration is required.

Note:: Dialer configuration is never required over a TCP/IP connection.

Permissions File

The **Permissions** file on system hera contains the following entry which grants system zeus access to system hera:

```
LOGNAME=uzeus SENDFILES=yes REQUEST=yes \
MACHINE=hera:zeus VALIDATE=zeus COMMANDS=rmail:who:uucp
```

This combined LOGNAME and MACHINE entry provides the following permissions to system zeus on system hera:

- System zeus can request and send files regardless of who initiated the call.
- System zeus can read and write only to the public directory (the default).
- System zeus can run only the **rmail**, **who**, and **uucp** commands.
- System zeus must log in to system hera as user uzeus and cannot use any other login ID for BNU transactions.

Note: Separately, the LOGNAME and MACHINE entries are:

```
LOGNAME=uzeus VALIDATE=zeus SENDFILES=yes REQUEST=yes
MACHINE=hera:zeus COMMANDS=rmail:who:uucp REQUEST=yes
```

BNU Configuration for a Telephone Connection Example

The following sample files are set up to connect systems venus and merlin over a telephone line using modems. System venus is considered the local system and system merlin the remote system.

On both systems, the device tty1 is hooked to a Hayes modem at 1200 baud. The login ID used for system venus to log into system merlin is uvenus, and the associated password is mirror. The login ID for system merlin to log into system venus is umerlin, and the associated password is oaktree. The phone number for the modem attached to venus is 9=3251436; the number of the merlin modem is 9=4458784. Both computers include partial phone numbers in their **Systems** files and dial–codes in their **Dialcodes** files.

Entries on the Local System

Files containing telephone connection entries on local system venus include the following:

Systems File

The **Systems** file on venus should contain the following entry for merlin, including a phone number and a dialing prefix:

```
merlin Any ACU 1200 local8784 ”” in:––in: uvenus word: mirror
```

System venus can call system merlin at any time, using an ACU device at 1200 baud and logging in as uvenus with the password mirror. The telephone number is expanded based on the code local in the **Dialcodes** file, and the device to be used is determined based on the *Type* and *Class* entries. Accordingly, BNU checks the **Devices** files for a device of type ACU and class 1200.

Dialcodes File

The **Dialcodes** file on system venus contains the following dial–code prefix for use with the number in the **Systems** file:

```
local 9=445
```

Given this code, the telephone number for system merlin in the **Systems** file is expanded to 9=4458784.

Devices File

The **Devices** file on system venus should contain the following entry for the connection to system merlin:

```
ACU tty1  –  1200  hayes \T
```

The port to be used is tty1, and the *Dialer* entry in the *Dialer–Token Pairs* field is hayes. The *Token* entry, \T, indicates that the telephone number is to be expanded using a code from the **Dialcodes** file. BNU checks the **Dialers** files for a hayes dialer type.

Dialers File

A **Dialers** file used by the **uucico** daemon on system venus should contain the following entry for the hayes modem:

```
hayes =,–, ”” \dAT\r\c OK \pATDT\T\r\c CONNECT
```

Note:: The expect–send characters are defined in the **Dialers** file format.

Permissions File

The **Permissions** file on system venus contains the following entries specifying the ways in which system merlin can conduct **uucico** and **uuxqt** transactions with system venus:

```
LOGNAME=umerlin REQUEST=yes SENDFILES=yes \
READ=/var/spool/uucppublic:/home/merlin \
WRITE=/var/spool/uucppublic:/home/merlin
MACHINE=venus:merlin VALIDATE=umerlin REQUEST=yes SENDFILES=yes    \
COMMANDS=ALL \
READ=/var/spool/uucppublic:/home/merlin \
WRITE=/var/spool/uucppublic:/home/merlin
```

System merlin logs in to system venus as umerlin, which is a unique login for system merlin. It can request and send files regardless of who initiated the call. Also, system merlin can read and write to the **/var/spool/uucppublic** directory and the **/home/merlin** directory on system venus. It can issue all commands in the default command set on system venus.

Entries on the Remote System

Files containing telephone connection entries on remote system merlin include the following:

Systems File

A **Systems** file on merlin should contain the following entry for venus, including a phone number and a dialing prefix:

```
venus Any ACU 1200 intown4362 ”” in:––in: umerlin word: oaktree
```

System merlin can call system venus at any time, using an ACU device at 1200 baud and logging in as user umerlin with the password oaktree. The telephone number is expanded based on the code intown in the **Dialcodes** file, and the device to be used is determined based on the *Type* and *Class* entries. Accordingly, BNU checks the **Devices** file(s) for a device of type ACU and class 1200.

Dialcodes File

The **Dialcodes** file on system merlin contains the following dial–code prefix for use with the number in the **Systems** file:

```
intown 9=325
```

Therefore, the expanded telephone number to reach system venus is 9=3254362.

Devices File

A **Devices** file on system merlin should contain the following entry for the connection to venus:

```
ACU  tty1  –  1200  hayes \T
```

The ACU is attached to port tty1, and the dialer is hayes. The telephone number is expanded with information from the **Dialcodes** file. BNU checks the **Dialers** files for an entry for a hayes modem.

Dialers File

A **Dialers** file used by the **uucico** daemon on system `merlin` should contain the following entry for its modem:

```
hayes  =,–,  ””  \dAT\r\c OK \pATDT\T\r\c CONNECT
```

Permissions File

The **Permissions** file on system `merlin` contains the following entries which grant system `venus` access to `merlin`:

```
LOGNAME=uvenus SENDFILES=call REQUEST=no \
WRITE=/var/spool/uucppublic:/home/venus \
READ=/var/spool/uucppublic:/home/venus
MACHINE=merlin:venus VALIDATE=uvenus  \
READ=/ WRITE=/ COMMANDS=ALL REQUEST=yes \
NOREAD=/etc/uucp:/usr/etc/secure \
NOWRITE=/etc/uucp:/usr/etc/secure
```

BNU Configuration for a Direct Connection Example

The following files are set up for a hardwired connection between systems `zeus` and `hera`, where `zeus` is considered the local system and `hera` the remote system. The hardwired device on system `zeus` is `tty5`; on system `hera` it is `tty1`. The speed of the connection is `1200` bps. The login ID for system `zeus` on system `hera` is `uzeus`, and the associated password is `thunder`. The login ID for system `hera` on system `zeus` is `uhera`, and the associated password is `portent`.

Entries in the Local System’s Files

Files containing telephone connection entries on local system `zeus` include the following:

Systems File

A **Systems** file on `zeus` should contain the following entry for the remote system `hera`:

```
hera Any hera 1200 – ”” \r\d\r\d\r in:––in: uzeus word:  thunder
```

This entry specifies that system `hera` can log in to system `zeus` at any time, using a direct connection specified in the **Devices** files. To find the entry in the **Devices** files, BNU uses the third and fourth fields of the **Systems** entry. Thus, BNU looks for an entry in the **Devices** files with a *Type* of `hera` and a *Class* of `1200`. System `zeus` logs in to system `hera` as user `uzeus` with the password `thunder`.

Devices File

A **Devices** file on `zeus` should contain the following entry in order to connect to the remote system `hera`:

```
hera    tty5   –  1200  direct
```

This entry specifies that system `zeus` uses the device `tty5` at `1200` bps to communicate with system `hera`. Note that the *Dialer* in both *Dialer–Token Pairs* fields is `direct`. When connecting to system `hera`, BNU checks the **Dialers** file for a `direct` entry.

Dialers File

A **Dialers** file on system `zeus` must contain the following entry for direct connections:

```
direct
```

This specifies that no handshaking is required on the direct connection.

Permissions File

The **Permissions** file on the local system `zeus` contains the following entry specifying the ways in which the remote system `hera` can conduct **uucico** and **uuxqt** transactions with `zeus`:

```
LOGNAME=uhera MACHINE=hera VALIDATE=uhera REQUEST=yes \
SENDFILES=yes MACHINE=zeus READ=/ WRITE=/ COMMANDS=ALL
```

This entry specifies that system `hera` logs in as `uhera`. Since the `VALIDATE=uhera` option is included, system `hera` cannot log in to system `zeus` with any other login ID, nor can any other remote system use the `uhera` ID. System `hera` can read and write to any directory on system `zeus`, and can send and request files regardless of who initiated the call. System `hera` can also initiate any commands on system `zeus`.

Note:: Since the permissions that are granted are the same regardless of which system initiated the connection, the LOGNAME and MACHINE entries have been combined. Separately, they are:

```
LOGNAME=uhera REQUEST=yes SENDFILES=yes READ=/ WRITE=/
MACHINE=zeus:hera VALIDATE=uhera READ=/ WRITE=/ REQUEST=yes \
COMMANDS=ALL
```

Attention: Providing the permissions in the preceding example is equivalent to giving any user on the remote system a login ID on the local system. Such liberal permissions can jeopardize your security and should usually be given only to well–trusted remote systems at the same site.

Entries in the Remote System Files

Files containing telephone connection entries on remote system `hera` include the following:

Systems File

A **Systems** file on system `hera` must contain the following entry for `zeus`:

```
zeus Any zeus 1200 – ”” \r\d\r\d\r in:––in: uhera word: portent
```

This entry specifies that system `hera` can log in to system `zeus` at any time, using a direct connection specified in the **Devices** files. To find the entry in the **Devices** files, BNU uses the third and fourth fields of the **Systems** entry. Thus BNU looks for an entry in the **Devices** files with a *Type* of `zeus` and a *Class* of `1200`. System `hera` logs in to system `zeus` as user `uhera` with the password `portent`.

Devices File

A **Devices** file on system `hera` must contain the following entry for communications with `zeus`:

```
zeus    tty1  –  1200  direct
```

This entry specifies that system `hera` uses the device `tty1` at `1200` bps to communicate with system `zeus`. Since the *Dialer* is specified as `direct`, BNU checks the **Dialers** files for a `direct` entry.

Dialers File

A **Dialers** file on system `hera` must contain the following entry for direct connections:

```
direct
```

This specifies that no dialer configuration is required on the direct connection.

Permissions File

The **Permissions** file on system hera contains the following entries specifying the ways zeus can conduct **uucico** and **uuxqt** transactions with hera:

```
LOGNAME=uzeus REQUEST=yes SENDFILES=yes READ=/ WRITE=/
MACHINE=hera:zeus VALIDATE=uzeus REQUEST=yes COMMANDS=ALL READ=/\
WRITE=/
```

These entries specify that system zeus logs in to system hera as uzeus. Because the VALIDATE=uzeus option is included, system zeus cannot log in to system hera with any other login ID, nor can any other remote system use the uzeus ID. System zeus can read and write to any directory on system hera, and can send and request files regardless of who initiated the call. System zeus can also initiate any commands on system hera.

Attention: Providing the permissions in the preceding example is equivalent to giving any user on the remote system a login ID on the local system. Such liberal permissions can jeopardize your security and normally should be given only to remote systems at the same site.

BNU Files, Commands, and Directories Reference

BNU Directories

/etc/uucp
Contains all Basic Network Utilities (BNU) configuration files.

/var/locks
Contains lock files for system devices. Used by other subsystems in addition to BNU.

/var/spool/uucppublic
Contains files that have been transferred by BNU.

/var/spool/uucp
Contains BNU administrative files.

/var/spool/uucp/.Workspace
Holds temporary files that the file transport programs use internally.

/var/spool/uucp/.Xqtdir
Contains execute files with lists of commands that remote systems can run.

/var/spool/uucp/ *SystemName*
Contains files used by file transport programs.

BNU Files

/etc/uucp/Systems
A list of systems to which **uucico** can connect.

/etc/uucp/Devices
Defines basic communication parameters for dial–out connections.

/etc/uucp/Permissions
Defines permissions for remote machines contacting the local machine through BNU.

Maxuuscheds
Limits simultaneous scheduled jobs.

Maxuuxqts
Limits simultaneous remote command executions.

/etc/uucp/Dialers
Specifies dialer and modem type.

/etc/uucp/Dialcodes
Contains the initial digits of telephone numbers used to establish remote connections over a phone line.

/usr/sbin/uucp/remote.unknown
A shell script executed when an unknown remote computer attempts to communicate.

/usr/sbin/uucp/Sysfiles
Assigns alternate or additional Systems, Devices, and Dialers files.

/etc/uucp/Poll
Determines when a remote system is called.

uudemon.admin
Sends a BNU status report to a specified login ID.

uudemon.cleanu
Cleans BNU spooling directories at prescheduled times.

uudemon.hour
Initiates file transport calls to remote systems.

uudemon.poll
Polls remote systems listed in the **/etc/uucp/Poll** file.

/var/spool/uucp/audit
Contains audit information from BNU activities.

/var/spool/uucp/Foreign Contains error information about BNU activities.

/var/spool/uucp/errors Contains error information about BNU activities.

/var/spool/uucp/xferstats Contains statistical information about BNU activities.

/var/spool/uucp/Corrupt Contains copies of files that cannot be processed by the BNU program.

/var/spool/uucp/.Log Contains log files from current BNU transactions.

/var/spool/uucp/.Old Contains log files from old BNU transactions.

/var/spool/uucp/.Status Stores the last time the **uucico** daemon tried to contact remote systems.

/var/spool/uucp/ *SystemName* **/C.*** These files are the commands allowed when connected to *SystemName*.

/var/spool/uucp/ *SystemName* **/D.*** These files are data files associated with *SystemName*.

/var/spool/uucp/ *SystemName* **/X.*** Executable files on *SystemName*.

/var/spool/uucp/ *SystemName* **/TM.*** Temporary files used when connected to *SystemName*.

BNU Commands

ct Connects to another system over a telephone line.

cu Connects to another system.

tip A variation of **cu** that requires special configuration.

uucp Copies files from one system to another running BNU or a version of the UNIX–to–UNIX Copy Program (UUCP).

uudecode Reconstructs a binary file encoded with **uuencode**.

uuencode Encodes a binary file into ASCII form for transmission using BNU.

uuname Provides information about accessible systems.

uupoll Forces a call to a remote system.

uuq Displays the BNU job queue.

uusend Sends a file to a remote host running BNU or UUCP.

uusnap Displays a brief summary of BNU status.

uustat Reports the status of BNU operations.

uuto Copies files to another system using BNU or UUCP.

uux Runs a command on a remote system.

uucheck Checks the **/etc/uucp/Permissions** file for correct configuration.

uuname Shows the names of all systems that can be reached through BNU.

uuclean Cleans BNU spooling directories.

uucleanup Cleans BNU spooling directories.

uukick Contacts a remote system with debugging enabled.

uulog Displays BNU log files.

uutry Contacts a remote system with debugging enabled; allows override of retry time.

uucpadm Administers the BNU system.

uupick Enables users to claim files in the **/var/spool/uucppublic** directory.

uucp Login ID with full administrative authority over the BNU subsystem.

Uutry Contacts a remote system with debugging turned on, and saves the debugging output in a file.

BNU Daemons

uucico Contacts remote systems and transfers files.

uucpd Allows BNU to run on top of Transmission Control Protocol/Internet Protocol (TCP/IP).

uusched Schedules BNU jobs.

uuxqt Runs command requests from remote systems.

Appendix A. PCI Adapters

This section provides installation and configuration information for PCI adapters. Topics discussed are support and configuration for the PCI Wide Area Network (WAN) Adapters (2–Port Multiprotocol HDLC Network Device Driver Overview on page A-1 and ARTIC960Hx PCI Adapter Overview on page A-2).

PCI Wide Area Network (WAN) Adapters

This section explains the installation and configuration requirements for the 2–Port Multiprotocol adapter (PCI), and the ARTIC960Hx PCI adapter.

2–Port Multiprotocol HDLC Network Device Driver Overview

The 2–Port Multiprotocol Adapter high–level data link control (HDLC) device driver is a component of the communication I/O subsystem. This device driver provides support for the HDLC operation over the 2–Port Multiprotocol adapter at speeds up to 1.544Mbps.

The following options provide access to the 2–Port Multiprotocol HDLC Network device driver:

- Systems Network Architecture (SNA)
- The synchronous data link control (SDLC) version of the GDLC Programming Interface
- User–written applications compatible with the SDLC MPQP–API (Multiprotocol Quad Port–Application Programming Interface)

Note:: The above options require use of the **mpc** *n* special file, which allows access to the 2–Port Multiprotocol adapter's HDLC device driver through the SDLC COMIO device driver emulation subsystem. This subsystem must be installed and configured for each HDLC network device.

- User–written applications compatible with the HDLC Common Data Link Interface (CDLI) API

The 2–port Multiprotocol Adapter device driver allows connectivity to remote host systems using the 2–port Multiprotocol adapter, either directly over a leased line or over switched circuits. The device driver can provide a gateway between work group environments and remote data processing facilities.

Configuring the 2–Port Multiprotocol Adapter

The following table explains how to configure a 2–Port Multiprotocol adapter.

Task	**SMIT fast path**	Web-based System Manager Management Environment [1]
Add a Device Driver to the Adapter	**smit mkhdlcdpmpdd**	
Reconfigure the Device Driver on the Adapter	**smit chhdlcdpmpdd**	
Remove a Device Driver on the Adapter	**smit rmhdlcdpmpdd**	
Make a Defined Device Driver Available	**smit cfghdlcdpmpdd**	

Add an SDLC COMIO Emulator on the Adapter	**smit mksdlcsciedd**	
Reconfigure the SDLC COMIO Emulator on the Adapter	**smit chsdlcsciedd**	
Remove an SDLC COMIO Emulator on the Adapter	**smit rmsdlcsciedd**	
Make a Defined SDLC COMIO Emulator Available	**smit cfgsdlcsciedd**	

Note:: These tasks are not available in the Web-based System Manager Management Environment.

ARTIC960Hx PCI Adapter Overview

The ARTIC960Hx PCI Adapter MPQP COMIO device driver emulator is a component of the communication I/O subsystem. This device driver provides support for the ARTIC960Hx PCI adapter at a maximum speed of 2M bps. The modems used must provide the clocking, since only external clocking is supported.

The following options provide access to the ARTIC960Hx PCI Adapter MPQP COMIO device driver:

- Systems Network Architecture (SNA)
- The generic data link control (GDLC) Programming Interface
- User–written applications compatible with the MPQP–API (Multiprotocol Quad Port–Application Programming Interface), such as SDLC and BiSync applications.

These options require use of the **mpq** *x* special file, which allows access to the ARTIC960Hx PCI adapter through the MPQP COMIO emulation device driver. This device driver must be installed and configured for each port on the ARTIC960Hx PCI adapter. The **mpq** *x* special file resides in the **/dev** directory.

Note:: The *x* in **mpq** *x* specifies the instance of the device driver; for example, `mpq0`.

The MPQP COMIO emulation device driver allows connectivity to remote host systems using the ARTIC960Hx PCI adapter, either directly over a leased line. The device driver can provide a gateway between work group environments and remote data processing facilities.

Configuring the MPQP COMIO Emulation Driver over the ARTIC960Hx PCI Adapter

The following table explains how to configure the MPQP COMIO Emulation Driver over the ARTIC960Hx PCI Adapter.

Task	**SMIT fast path**	Web-based System Manager Management Environment [1]
Add a Device Driver	**smit mktsdd**	
Reconfigure the MPQP COMIO Emulation Driver	**smit chtsdd**	
Remove a Device Driver	**smit rmtsdd**	
Configure a Defined Device Driver	**smit cfgtsdd**	
Add a Port	**smit mktsdports**	
Reconfigure an MPQP COMIO Emulation Port	**smit chtsdports**	

Remove a Port	**smit rmtsdports**	
Configure a Defined Port	**smit cfgtsdports**	
Trace the MPQP COMIO Emulation Driver	**smit trace_link**	

Note:: These tasks are not available in the Web-based System Manager Management Environment.

Appendix B. Configure Network Interface Backup in previous AIX versions

These instructions lead you through the steps on setting up an EtherChannel in AIX 4.3.3 and AIX 5.1. For more information about EtherChannel, or for instructions on how to perform this task in AIX 5.2 and later, see EtherChannel and IEEE 802.3ad Link Aggregation on page 4-197.

1. With root authority, type `smit etherchannel` on the command line.
2. Select **Add an Etherchannel** and press Enter.
3. Select the adapters that you want to include in the EtherChannel, both primary and secondary.

   Note:: The **Available Network Adapters** displays all Ethernet adapters. If you select an Ethernet adapter that is already being used, you will get an error message. You first need to detach that interface by using the **ifconfig** command.
4. Enter the information in the fields according to the following guidelines:
   - **Etherchannel adapters**: You should see the adapters that you selected in the previous step.
   - **Enable ALTERNATE ETHERCHANNEL address**: This field is optional. Setting this to `yes` will enable you to specify a MAC address that you want the EtherChannel to use. If you set this option to `no`, the EtherChannel will use the MAC address of the first adapter specified.
   - **ALTERNATE ETHERCHANNEL address**: If you set **Enable ALTERNATE ETHERCHANNEL address** to `yes`, specify the MAC address that you want the EtherChannel to use here. The address you specify must start with `0x` and be a 12–digit hexadecimal value.
   - **Mode**: Select `netif_backup`.
   - **Enable GIGABIT ETHERNET JUMBO frames**: This field is optional. In order to use this, your switch must support jumbo frames. This will only work on Standard Ethernet, not IEEE 802.3. Set this to `yes` if you want to enable it.
   - **Internet Address to Ping**: This field is optional. The EtherChannel will ping the IP address you specify here. If the EtherChannel is unable to ping this address for **Number of Retries** times in **Retry Timeout** intervals, the EtherChannel will switch adapters.
   - **Number of Retries**: Enter the number of ping response failures that are allowed before the EtherChannel switches adapters. The default is three. This field is optional and valid only if you set an **Internet Address to Ping**.
   - **Retry Timeout**: Enter the number of seconds between the times when the EtherChannel will ping the **Internet Address to Ping**. The default is one second. This field is optional and valid only if you have set and **Internet Address to Ping**.
5. Press Enter after changing the desired fields to create the EtherChannel.
6. Configure IP over the new interface by typing `smit chinet` at the command line.
7. Select your new EtherChannel interface from the list.
8. Fill in all required fields and press Enter.

Appendix C. Conversion Table

ASCII	Decimal	Hexadecimal	Octal	Binary
null	0	0	0	0
start of header	1	1	1	1
start of text	2	2	2	10
end of text	3	3	3	11
end of transmission	4	4	4	100
enquire	5	5	5	101
acknowledge	6	6	6	110
bell	7	7	7	111
backspace	8	8	10	1000
horizontal tab	9	9	11	1001
linefeed	10	A	12	1010
vertical tab	11	B	13	1011
form feed	12	C	14	1100
carriage return	13	D	15	1101
shift out	14	E	16	1110
shift in	15	F	17	1111
data link escape	16	10	20	10000
device control 1/Xon	17	11	21	10001
device control 2	18	12	22	10010
device control 3/Xoff	19	13	23	10011
device control 4	20	14	24	10100
negative acknowledge	21	15	25	10101
synchronous idle	22	16	26	10110
end of transmission block	23	17	27	10111
cancel	24	18	30	11000
end of medium	25	19	31	11001
end of file/ substitute	26	1A	32	11010
escape	27	1B	33	11011
file separator	28	1C	34	11100
group separator	29	1D	35	11101

record separator	30	1E	36	11110
unit separator	31	1F	37	11111
space	32	20	40	100000
!	33	21	41	100001
”	34	22	42	100010
#	35	23	43	100011
$	36	24	44	100100
%	37	25	45	100101
&	38	26	46	100110
’	39	27	47	100111
(	40	28	50	101000
)	41	29	51	101001
*	42	2A	52	101010
+	43	2B	53	101011
,	44	2C	54	101100
–	45	2D	55	101101
.	46	2E	56	101110
/	47	2F	57	101111
0	48	30	60	110000
1	49	31	61	110001
2	50	32	62	110010
3	51	33	63	110011
4	52	34	64	110100
5	53	35	65	110101
6	54	36	66	110110
7	55	37	67	110111
8	56	38	70	111000
9	57	39	71	111001
:	58	3A	72	111010
;	59	3B	73	111011
<	60	3C	74	111100
=	61	3D	75	111101
>	62	3E	76	111110
?	63	3F	77	111111
@	64	40	100	1000000
A	65	41	101	1000001
B	66	42	102	1000010
C	67	43	103	1000011
D	68	44	104	1000100
E	69	45	105	1000101
F	70	46	106	1000110
G	71	47	107	1000111

H	72	48	110	1001000
I	73	49	111	1001001
J	74	4A	112	1001010
K	75	4B	113	1001011
L	76	4C	114	1001100
M	77	4D	115	1001101
N	78	4E	116	1001110
O	79	4F	117	1001111
P	80	50	120	1010000
Q	81	51	121	1010001
R	82	52	122	1010010
S	83	53	123	1010011
T	84	54	124	1010100
U	85	55	125	1010101
V	86	56	126	1010110
W	87	57	127	1010111
X	88	58	130	1011000
Y	89	59	131	1011001
Z	90	5A	132	1011010
[	91	5B	133	1011011
\	92	5C	134	1011100
]	93	5D	135	1011101
^	94	5E	136	1011110
_	95	5F	137	1011111
‘	96	60	140	1100000
a	97	61	141	1100001
b	98	62	142	1100010
c	99	63	143	1100011
d	100	64	144	1100100
e	101	65	145	1100101
f	102	66	146	1100110
g	103	67	147	1100111
h	104	68	150	1101000
i	105	69	151	1101001
j	106	6A	152	1101010
k	107	6B	153	1101011
l	108	6C	154	1101100
m	109	6D	155	1101101
n	110	6E	156	1101110
o	111	6F	157	1101111
p	112	70	160	1110000
q	113	71	161	1110001
r	114	72	162	1110010

s	115	73	163	1110011
t	116	74	164	1110100
u	117	75	165	1110101
v	118	76	166	1110110
w	119	77	167	1110111
x	120	78	170	1111000
y	121	79	171	1111001
z	122	7A	172	1111010
{	123	7B	173	1111011
|	124	7C	174	1111100
}	125	7D	175	1111101
~	126	7E	176	1111110
DEL	127	7F	177	1111111
	128	80	200	10000000
	129	81	201	10000001
	130	82	202	10000010
	131	83	203	10000011
	132	84	204	10000100
	133	85	205	10000101
	134	86	206	10000110
	135	87	207	10000111
	136	88	210	10001000
	137	89	211	10001001
	138	8A	212	10001010
	139	8B	213	10001011
	140	8C	214	10001100
	141	8D	215	10001101
	142	8E	216	10001110
	143	8F	217	10001111
	144	90	220	10010000
	145	91	221	10010001
	146	92	222	10010010
	147	93	223	10010011
	148	94	224	10010100
	149	95	225	10010101
	150	96	226	10010110
	151	97	227	10010111
	152	98	230	10011000
	153	99	231	10011001
	154	9A	232	10011010
	155	9B	233	10011011
	156	9C	234	10011100
	157	9D	235	10011101

	158	9E	236	10011110
	159	9F	237	10011111
	160	A0	240	10100000
	161	A1	241	10100001
	162	A2	242	10100010
	163	A3	243	10100011
	164	A4	244	10100100
	165	A5	245	10100101
	166	A6	246	10100110
	167	A7	247	10100111
	168	A8	250	10101000
	169	A9	251	10101001
	170	AA	252	10101010
	171	AB	253	10101011
	172	AC	254	10101100
	173	AD	255	10101101
	174	AE	256	10101110
	175	AF	257	10101111
	176	B0	260	10110000
	177	B1	261	10110001
	178	B2	262	10110010
	179	B3	263	10110011
	180	B4	264	10110100
	181	B5	265	10110101
	182	B6	266	10110110
	183	B7	267	10110111
	184	B8	270	10111000
	185	B9	271	10111001
	186	BA	272	10111010
	187	BB	273	10111011
	188	BC	274	10111100
	189	BD	275	10111101
	190	BE	276	10111110
	191	BF	277	10111111
	192	C0	300	11000000
	193	C1	301	11000001
	194	C2	302	11000010
	195	C3	303	11000011
	196	C4	304	11000100
	197	C5	305	11000101
	198	C6	306	11000110
	199	C7	307	11000111
	200	C8	310	11001000

	201	C9	311	11001001
	202	CA	312	11001010
	203	CB	313	11001011
	204	CC	314	11001100
	205	CD	315	11001101
	206	CE	316	11001110
	207	CF	317	11001111
	208	D0	320	11010000
	209	D1	321	11010001
	210	D2	322	11010010
	211	D3	323	11010011
	212	D4	324	11010100
	213	D5	325	11010101
	214	D6	326	11010110
	215	D7	327	11010111
	216	D8	330	11011000
	217	D9	331	11011001
	218	DA	332	11011010
	219	DB	333	11011011
	220	DC	334	11011100
	221	DD	335	11011101
	222	DE	336	11011110
	223	DF	337	11011111
	224	E0	340	11100000
	225	E1	341	11100001
	226	E2	342	11100010
	227	E3	343	11100011
	228	E4	344	11100100
	229	E5	345	11100101
	230	E6	346	11100110
	231	E7	347	11100111
	232	E8	350	11101000
	233	E9	351	11101001
	234	EA	352	11101010
	235	EB	353	11101011
	236	EC	354	11101100
	237	ED	355	11101101
	238	EE	356	11101110
	239	EF	357	11101111
	240	F0	360	11110000
	241	F1	361	11110001
	242	F2	362	11110010
	243	F3	363	11110011

	244	F4	364	11110100
	245	F5	365	11110101
	246	F6	366	11110110
	247	F7	367	11110111
	248	F8	370	11111000
	249	F9	371	11111001
	250	FA	372	11111010
	251	FB	373	11111011
	252	FC	374	11111100
	253	FD	375	11111101
	254	FE	376	11111110
	255	FF	377	11111111

Index

Vos remarques sur ce document / Technical publication remark form

Titre / Title : Bull System Management Guide Communications and Networks

Nº Reférence / Reference Nº : 86 A2 27EF 03

Daté / Dated : June 2003

ERREURS DETECTEES / ERRORS IN PUBLICATION

AMELIORATIONS SUGGEREES / SUGGESTIONS FOR IMPROVEMENT TO PUBLICATION

Vos remarques et suggestions seront examinées attentivement.
Si vous désirez une réponse écrite, veuillez indiquer ci-après votre adresse postale complète.

Your comments will be promptly investigated by qualified technical personnel and action will be taken as required.
If you require a written reply, please furnish your complete mailing address below.

NOM / NAME : ______________________________ Date : __________

SOCIETE / COMPANY : ______________________________

ADRESSE / ADDRESS : ______________________________

____

Remettez cet imprimé à un responsable BULL ou envoyez-le directement à :

Please give this technical publication remark form to your BULL representative or mail to:

BULL CEDOC
357 AVENUE PATTON
B.P.20845
49008 ANGERS CEDEX 01
FRANCE

Technical Publications Ordering Form

Bon de Commande de Documents Techniques

To order additional publications, please fill up a copy of this form and send it via mail to:
Pour commander des documents techniques, remplissez une copie de ce formulaire et envoyez-la à :

BULL CEDOC
ATTN / Mr. L. CHERUBIN
357 AVENUE PATTON
B.P.20845
49008 ANGERS CEDEX 01
FRANCE

Phone / Téléphone :	+33 (0) 2 41 73 63 96
FAX / Télécopie	+33 (0) 2 41 73 60 19
E–Mail / Courrier Electronique :	srv.Cedoc@franp.bull.fr

Or visit our web sites at: / Ou visitez nos sites web à:

http://www.logistics.bull.net/cedoc

http://www–frec.bull.com   http://www.bull.com

CEDOC Reference # N° Référence CEDOC	**Qty** Qté	**CEDOC Reference #** N° Référence CEDOC	**Qty** Qté	**CEDOC Reference #** N° Référence CEDOC	**Qty** Qté
__ __ ____ _ [__]		__ __ ____ _ [__]		__ __ ____ _ [__]	
__ __ ____ _ [__]		__ __ ____ _ [__]		__ __ ____ _ [__]	
__ __ ____ _ [__]		__ __ ____ _ [__]		__ __ ____ _ [__]	
__ __ ____ _ [__]		__ __ ____ _ [__]		__ __ ____ _ [__]	
__ __ ____ _ [__]		__ __ ____ _ [__]		__ __ ____ _ [__]	
__ __ ____ _ [__]		__ __ ____ _ [__]		__ __ ____ _ [__]	
__ __ ____ _ [__]		__ __ ____ _ [__]		__ __ ____ _ [__]	
[_ _] : **no revision number means latest revision** / pas de numéro de révision signifie révision la plus récente					

NOM / NAME : ____ Date : ____________

SOCIETE / COMPANY : ____

ADRESSE / ADDRESS : ____

____

PHONE / TELEPHONE : ____________________ FAX : ____________________

E–MAIL : ____

For Bull Subsidiaries / Pour les Filiales Bull :

Identification: ____

For Bull Affiliated Customers / Pour les Clients Affiliés Bull :

Customer Code / Code Client : ____

For Bull Internal Customers / Pour les Clients Internes Bull :

Budgetary Section / Section Budgétaire : ____

For Others / Pour les Autres :

Please ask your Bull representative. / Merci de demander à votre contact Bull.

PLACE BAR CODE IN LOWER
LEFT CORNER

BULL CEDOC
357 AVENUE PATTON
B.P.20845
49008 ANGERS CEDEX 01
FRANCE

ORDER REFERENCE
86 A2 27EF 03

Bull

Utiliser les marques de découpe pour obtenir les étiquettes.
Use the cut marks to get the labels.

AIX

System
Management
Guide
Communications
and Networks

86 A2 27EF 03

AIX

System
Management
Guide
Communications
and Networks

86 A2 27EF 03

AIX

System
Management
Guide
Communications
and Networks

86 A2 27EF 03