NetView® for UNIX®

Configuration Guide

Version 7

SC31-8894-00

Tivoli NetView for UNIX Configuration Guide

Copyright Notice

Trademarks

Notices

Contents

Preface

This document provides information about starting, stopping, configuring, and maintaining the Tivoli® NetView program.

When referring to the host connection, this book assumes you are connecting to Tivoli NetView for OS/390®.

Who Should Read This Guide

This book is designed for system administrators and network operators who are familiar with the operation of networks. Anyone involved in configuring and maintaining the Tivoli NetView program should read this book.

This book assumes that the user has a general understanding of network management and of how the Tivoli NetView program fits into that environment. An understanding of the AIX® or Solaris operating system is required to configure the Tivoli NetView program.

Prerequisite and Related Documents

The following is a list of Tivoli NetView related publications:

Tivoli NetView Administrator's Guide
Tivoli NetView Administrator's Reference
Tivoli NetView Database Guide
Tivoli NetView Host Connection
Tivoli NetView Configuration Guide
Tivoli NetView MLM User's Guide
Tivoli NetView Programmer's Guide
Tivoli NetView Programmer's Reference
Tivoli NetView User's Guide for Beginners
TME 10™ Framework Reference Manual

What This Guide Contains

This book is organized by task. Each chapter contains a task, or tasks, and the steps required to complete that task or tasks:

- "Chapter 1. Preparing to Use Tivoli NetView" on page 1

  Describes tasks to perform before starting Tivoli NetView, including migrating relational database information.

- "Chapter 2. Installing and Using the AIX trapgend Daemon" on page 5

  Describes the steps for installing the trapgend daemon on remote RS/6000® nodes and other operations available after installing the trapgend daemon, such as adding and deleting trap destinations on remote nodes.

- "Chapter 3. Uninstalling Tivoli NetView" on page 11

  Describes how to deinstall Tivoli NetView, including deinstalling clients, servers, the trapgend daemon, and the mid-level manager.

- "Chapter 4. Starting and Stopping Tivoli NetView" on page 13

  Describes the processes for starting and stopping the Tivoli NetView program. This includes starting and customizing the netview shell, defining the network

management region, and setting up and configuring servers and clients. This chapter also provides the steps for starting and stopping the daemons and restarting map generation.

- "Chapter 5. Optional Configuration Tasks" on page 29

  Describes additional configuration options you can apply to the Tivoli NetView program. You can change the defaults for the daemons, or perform other optional configuration tasks. For example, you can add entries to the object identification files (oid_to_type, oid_to_sym, or oid_to_command) anytime after installation.

- "Chapter 6. Maintaining Tivoli NetView" on page 61

  Describes on-going maintenance tasks you can do to optimize the performance of the Tivoli NetView program.

The glossary at the end of this document can assist you with terminology. To view additional terminology lists, refer to:

`http://www-3.ibm.com/ibm/terminology/`

Typeface Conventions

This guide uses several typeface conventions for special terms and actions. These conventions have the following meaning:

Bold Commands, keywords, file names, authorization roles, URLs, or other information that you must use appear in **bold**. The names or titles of screen objects also appear in **bold**.

Italics Variables and values that you must provide appear in *italics*.Words and phrases that are emphasized also appear in *italics*.

Bold Italics
New terms appear in ***bold italics*** when they are defined in text.

`Monospace`
Code examples, output and system messages appear in a `monospace` font.

`ALL CAPS`
Tivoli NetView for OS/390 commands appear in `ALL CAPS`.

Platform-Specific Information

Refer to the release notes for platform-specific information.

Online Information

The release notes provide the latest information on the Tivoli NetView program. They are available in HTML and PDF versions. The HTML version is accessible from the NetView Console using the **Help...Books Online** menu item. The PDF version is in /usr/OV/books/$LANG/pdf/readme.pdf.

The online help facility provides task and user interface information.

The online books are available in HTML and PDF versions (Dynatext is no longer supported). The HTML versions are accessible from the NetView Console using the **Help...Books Online** menu item, which will bring up the books in the Netscape Navigator or Netscape Communicator browser.

PDF versions are available in the /usr/OV/books/$LANG/pdf directory.

In addition, you can access online documents at this web site:

`http://www.tivoli.com/support`

A user name and password are required.

Accessability Information

Refer to *Tivoli NetView for UNIX User's Guide for Beginners* for information about accessability.

Keyboard Access

Standard shortcut and accelerator keys are used by the product and are documented by the operating system. Refer to the documentation provided by your operating system for more information.

Refer to *Tivoli NetView for UNIX User's Guide for Beginners* for more information about keyboard access.

Contacting Tivoli Support

If you have a problem with any Tivoli product, you can contact Tivoli Customer Support. See the *Tivoli Customer Support Handbook* at the following Web site:

http://www.tivoli.com/support/handbook/

The handbook provides information about how to contact Tivoli Customer Support, depending on the severity of your problem, and the following information:

- Registration and eligibility
- Telephone numbers and e-mail addresses, depending on the country you are in
- What information you should gather before contacting support

Chapter 1. Preparing to Use Tivoli NetView

This chapter describes tasks that you should complete before starting Tivoli NetView. This includes:

- "Renaming and Deleting Files"
- "Migrating Relational Database Information"
- "Additional Information" on page 2

Renaming and Deleting Files

You may want to rename or delete any existing files from a previous installation, such as Version 6 migration files (/usr/OV.back.v6r0). Deleting or renaming these files prevents data from being migrated if you decide to reinstall Version 7. Deleting the files also saves file system space.

Note: It is a good idea to tar the backup directory to tape or another archive medium. You might need it at a later date.

Migrating Relational Database Information

If you have configured your previous NetView installation to use relational database support which you want to use with your Tivoli NetView Version 7 installation, follow these steps after installing Tivoli NetView.

Migrating from NetView Version 5.1 or 6.0

By default, relational database management systems (RDBMSs) are not used to store data. If you have configured your NetView Version 5.1 installation to use a relational database to store IP topology, SNMP collection, or trapd log data, and you want to save this information to be used by your Tivoli NetView Version 7 installation, migrate the information.

The following steps are required to migrate the relational database information:

1. Create the RDBMS Interface Module (RIM) object by following the instructions in the *Tivoli NetView Database Guide*. Note that you will need to re-create the RIM object.
2. Using the charts in Chapter 6 of the *Tivoli NetView Database Guide*, ensure that tables with a column size of 251 contain no more than 251 bytes. If they do, edit them down to 251 bytes (this is because the database tables' maximum column length was changed from 254 to 251 bytes in Version 5.1.1).
3. If you are migrating from Version 5.1, transfer topology, trapd, and snmpCollect data from the relational database to flat files. If you are migrating from Version 6.0, transfer snmpCollect data from the relational database to flat files.
4. Clear the relational database tables.
5. Using the new scripts installed by the Tivoli NetView database component in **/usr/OV/scripts**, run the appropriate scripts to create new database tables for the types of data to be stored as follows:
   - For Version 5.1, run the topology, trapd, and snmpCollect scripts.
   - For Version 6.0, run the snmpCollect script.

   These scripts will automatically drop the old schema before creating the new one.
6. Transfer the flat file data to the (new) relational database.

Refer to the *Tivoli NetView Database Guide* for specific information on how to perform the steps outlined above.

Additional Information

The installation procedure adds entries for the Tivoli NetView processes to the appropriate files. These entries should not be changed. See “Installation Entries” on page 86 for a description of the installation entries.

Configuring the Tivoli NetView program is not necessary; you can start the Tivoli NetView program using the defaults provided. However, you can modify the defaults if you choose. See “Chapter 5. Optional Configuration Tasks” on page 29 for information about Tivoli NetView configuration options, including daemon options.

If you want the initial map to include all the networks in your administrative domain, or if you have a large network, you might want to preconfigure your system. For example, you might want to start netmon with a seed file. See “Using a Seed File to Customize Discovery” on page 38 for information about starting the netmon daemon with a seed file.

To increase or decrease the amount of memory used by the ovwdb daemon, see “Topology Discovery and Database Daemons” on page 31.

Configuring Client/Server Access

If you have installed Tivoli NetView clients, configure the server and clients in the following order:

1. Configure the server to enable the clients to access it, as described in “Configuring a Server to Enable Client Access”.
2. Configure the client to access the server, as described in “Configuring a Client to Access a Server”.

Configuring a Server to Enable Client Access

Multiple clients can access a server simultaneously. You must configure the server to enable access for each client. To add client access, complete the following steps:

1. Enter **/usr/OV/bin/serversetup** from the command line to start the Tivoli NetView Server Setup application.
2. Select **Configure –> Configure Tivoli NetView Client –> Add Client Access**. The **Add Client Access** dialog is displayed.
3. Enter the name of the client and click **OK**. The results of the command appear in the **Output** window.

If the server and client reside in different domains, the /etc/hosts file might need an additional entry to assist with communications between the server and client.

Configuring a Client to Access a Server

A client can only access one server at a time. Ensure that you have granted the client access to the server before configuring the client. If you have not, complete the steps in the “Configuring a Server to Enable Client Access” section. To configure a client to communicate with a server, complete the following steps on the client machine:

1. Ensure that the date and time on the client machine are the same date and time as are on the server machine. Use the **date** command to check this. The date and time must be synchronized for security and map administration to work correctly.
2. Enter **/usr/OV/bin/clientsetup** from the command line to start the Tivoli NetView Client Setup application.
3. Select **Configure –> Add/Change Server**. The **Add/Change Server** dialog is displayed.
4. Enter the name of the server. Select the location for the Map database. This value determines whether the map database will reside locally on the client machine or be NFS mounted from the server.

At the time a client is configured to access a server, NFS mounts are performed from the client to the server for the following:

- /usr/OV/conf
- /usr/OV/databases/snmpCollect

If the map database location is set to NFS, then NFS mounts from the server are also performed for the following:

- /usr/OV/databases/openview/mapdb
- /usr/OV/databases/openview/defmap

Refer to the release notes for more information.

Chapter 2. Installing and Using the AIX trapgend Daemon

The trapgend daemon is used only with the Tivoli NetView program that runs on an AIX operating system.

Install the latest level of the trapgend daemon on all remote RS/6000 nodes. Installing the trapgend daemon on all remote RS/6000 nodes provides additional management capabilities by:

- Enabling remote ping
- Enabling CPU use and disk space monitoring
- Converting AIX alertable errors to SNMP traps

You can find more information on the trapgend daemon than this section provides by referring to the *Tivoli NetView MLM User's Guide.*

Understanding the trapgend Daemon

The trapgend daemon is a subagent provided with the Tivoli NetView server program.

The trapgend daemon converts alertable errors generated by a remote RS/6000 node to SNMP traps and sends them to the Tivoli NetView management system. The traps can be found in two places. On the agent, the trap can be found in system error log (errpt -c). On the management system, the trap can be found in the trapd.log.

Note: To include failing hardware information in the alerts, you must install the Product Topology Data diskette on the remote RS/6000 node. This diskette contains vital product data for your system unit. For information about installing the Product Topology Data diskette, refer to the documentation shipped with your system unit.

The installation process adds an error notification object for the trapgend daemon to the Object Data Management (ODM) database and automatically starts the trapgend daemon and the AIX SNMP agent, snmpd, on the remote node.

You can install the trapgend daemon and access trapgend operations using the following methods:

- Using the Tivoli desktop

  Using the Tivoli desktop for trapgend daemon operations enables you to perform one trapgend operation on one remote node at a time.

  See "Installing and Configuring trapgend Using the Tivoli desktop" on page 6 for information about using Tivoli NetView to access trapgend operations.
- Using the nv6000_smit shell script

  Using the nv6000_smit shell script enables you to perform multiple trapgend operations on a remote node.

  See "Installing and Configuring trapgend Using a Shell Script" on page 8 for information about using the nv6000_smit shell script to access trapgend operations.

A root password is required with either method.

For AIX Only

Use the Tivoli desktop or the nv6000_smit shell script to perform trapgend operations. However, if you use SMIT, you should be aware that when you enter the password in the SMIT dialog, the password is written in the smit.log and smit.script files, which anyone can read. If you leave the SMIT password field blank, you will be prompted for the password, and the password will not be written in the smit.log or smit.script files.

With either method, you can perform the following trapgend operations:

- Install or update the trapgend daemon.
- Add and delete trap destinations.
- Start and stop the trapgend daemon.
- Check the status and test the trapgend daemon.
- Remove the trapgend daemon.

Note: You can remove the trapgend daemon from a remote node only if the Tivoli NetView server program is not installed on the remote node.

Installing and Configuring trapgend Using the Tivoli desktop

You must have root permissions to perform any of the trapgend operations. To access trapgend operations using the Tivoli desktop, follow this procedure:

1. To access the Tivoli desktop, enter **tivoli** on the command line.
2. Click and hold down the right mouse button on the server icon to display the icon's pull-down menu.
3. Select **Configure –> Install/configure subagent (trapgend) on remote system**.

   The Install/configure subagent (trapgend) on system dialog is displayed.

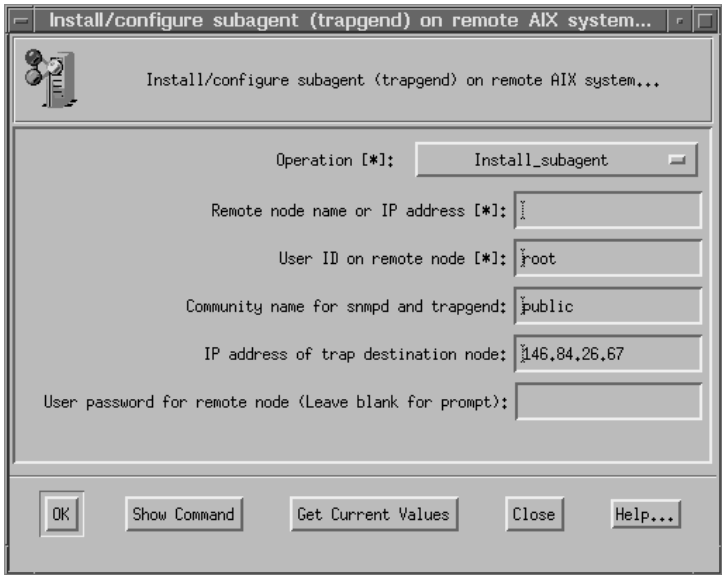

Figure 1. Install/Configure Subagent (trapgend) on Remote Node Dialog

4. Make the necessary changes in the required entry fields. See Table 1 for more detail about the entry fields.

Table 1. trapgend Options

Option	Action
Remote Operation	Select one of the following operations you want to perform on the remote node: **Add trap destination only** Add a trap destination on a remote node. **Delete trap destination only** Delete a trap destination on a remote node. **Install subagent** Install the trapgend daemon on a remote node. **Remove subagent** Remove the trapgend daemon from a remote node. The Tivoli NetView program cannot be installed on the remote node. **Start subagent** Start the trapgend daemon without any other options. **Status of subagent** View status of the trapgend daemon and trap destinations. **Stop subagent** Stop the trapgend daemon. **Test subagent** Test the operation of the trapgend daemon. **Update subagent** Update the remote trapgend key files to the same level at the server.
Remote node name or IP address	Enter the node name or IP address of the remote node.
User ID on remote node	Enter the user ID on the remote node. The default is **root**, but you can specify a user ID that has the appropriate permissions (a member of the system group, group 0).
Community name for snmpd and trapgend	Enter the community name of the remote node. The default is **public**.
IP address of trap destination node	Enter the IP address of the manager node set to receive traps from the remote node (if different than the default provided).
User password for remote node	Enter the password for the user ID on the remote node. If you type the password here, it will be displayed as you type it. If you leave this field blank, the program will prompt you for a password, and it will not be displayed as you type it.

5. Click **OK**.
6. Do one of the following:
   - If you entered a password in the User password for remote node field of the Install/configure subagent dialog, go to Step 7. The entries will be processed and the latest trapgend will be installed on the remote node.
   - If you did not enter a password, enter a password when the program prompts you for it. The password will not be displayed as you type it, and the cursor will not move as you type. The entries will be processed and the latest trapgend will be installed on the remote node.
7. Click **Close**.

   The dialog closes.

Installing and Configuring trapgend Using a Shell Script

You can create a shell script to call the /usr/OV/bin/nv6000_smit shell script to perform multiple trapgend operations on remote nodes. You must have root permissions to use the /usr/OV/bin/nv6000_smit shell script.

Note: The /usr/OV/bin/nv6000_smit shell script requires a password for the user ID you specify in your shell script. If you do not want to include the password in your shell script, run the shell script in the foreground, and you will be prompted to enter the password. If you want your shell script to be able to run unattended, you must include the password in your shell script.

To use the /usr/OV/bin/nv6000_smit shell script for multiple operations, follow this procedure:

1. Create a shell script.
2. Add a new line for each operation you want the shell script to perform. See "Example of a Shell Script" for an example of lines in the netview shell script. Each line in the shell script must call the /usr/OV/bin/nv6000_smit shell script and include the following parameters:
   - Keyword (subagentR)
   - Operation to perform (install, update, status, start, test, stop, addtrap, deletetrap, or remove)
   - Remote node name or IP address
   - User ID on the remote node (root or a user ID with the appropriate permissions, a member of the system group, group 0)
   - Community name
   - Trap destination
   - Password for the user ID on the remote node (optional). If you do not specify a password, you will be prompted to enter one when your shell script is run.
3. Save and execute the shell script file.

Example of a Shell Script

The following example shows lines in a shell script created for multiple trapgend daemon operations. The first line adds a trap destination for a remote node. The second line installs the trapgend daemon on a remote node.

```
/usr/OV/bin/nv6000_smit subagentR addtrap mlsnm003 userID \
  public 9.67.5.189

/usr/OV/bin/nv6000_smit subagentR install mdcnm008 userID \
  public 9.67.163.41 password
```

The variables indicate the following:

subagentR
Keyword

addtrap
Remote operation

mlsnm003
Remote node name

userID
User ID on the remote node (root or the user ID with the appropriate permissions, a member of the system group, group 0)

public Community name

9.67.5.189
IP address of the manager node to receive traps

password
Password for the user ID on the remote node

Chapter 3. Uninstalling Tivoli NetView

This chapter describes how to remove Tivoli NetView from your system. It provides steps to remove Tivoli NetView components and databases. Uninstalling Tivoli NetView will remove all Tivoli NetView directories and data.

To save the NetView databases or any other customized data before uninstalling, see “Saving Files” on page 85.

Uninstalling a Client

When you uninstall a client, Tivoli NetView does the following:

- Removes the client code from the client machine.
- Removes the NFS mount connections.

> **For AIX Only**
> When you uninstall a client, the trapgend subagent is not removed. If you want to remove the trapgend subagent, you must do it separately. Refer to “Uninstalling trapgend from a Remote Node (AIX only)” on page 12.

You must have root permissions to uninstall a client.

Note: If the client you are removing has local maps, delete those maps using the graphical user interface on the client before you remove the client code. Refer to the *Tivoli NetView for UNIX Administrator's Guide* for instructions. If you do not remove the client's local maps, the server's object database will contain incorrect information about the number and the location of maps that exist.

Complete the following steps:

1. Enter **/usr/OV/bin/serversetup** from the command line to start the Tivoli NetView Server Setup application.
2. Select **Configure –> Remove Server**.

   The Remove Server dialog is displayed. It is necessary to remove the client access to the server before uninstalling the client.
3. Make the necessary changes to the dialog fields and click **OK**. Refer to the online help for information about these fields.
4. Select **Maintain –> Deinstall Tivoli NetView Client**. The Deinstall Tivoli NetView Client dialog is displayed.
5. Change the dialog fields and click **OK**. Refer to the online help for information about these fields.
6. Click **OK** on the verification message dialog to start the removal. The Tivoli NetView code is removed from the client machine.
7. Click **X** on the Server Setup application window when deinstallation is complete to shut down the Server Setup Application.

Uninstalling a Server

Complete the following steps to remove all Tivoli NetView directories and data:

1. Exit the Tivoli NetView graphical user interfaces on this server and any clients or applications that reference this server.
2. Enter **/usr/OV/bin/serversetup** from the command line to start the Tivoli NetView Server Setup application.
3. Select **Maintain –> Deinstall Tivoli NetView**.
4. Select either **Deinstall Tivoli NetView (if no dependent products installed)** or **Deinstall Tivoli NetView (leave dependent products installed)**, depending on whether you want to uninstall Tivoli NetView even if there are dependent products installed. Refer to the online help for information on these two options.
5. Click **OK** on the verification message dialog to start the removal. The Tivoli NetView code is removed from the server machine.
6. Click **X** on the Server Setup application window when removal is complete to shut down the Server Setup Application.

Uninstalling trapgend from a Remote Node (AIX only)

You can remove the trapgend daemon from a remote node only if the Tivoli NetView server is not installed on the remote node. Complete the following steps to remove trapgend from a remote node:

1. Enter `/usr/OV/bin/serversetup` from the command line to start the Tivoli NetView Server Setup application.
2. Select **Configure –> Install/configure subagent (trapgend) on remote RISC System/6000**.
3. Select **Remove subagent** in the **Remote Operation** field.
4. Make the necessary changes to the dialog fields and click **OK**. Refer to the online help for information about these fields.

Uninstalling the Mid-Level Manager

Refer to the *Tivoli NetView Mid-Level Manager User's Guide* for information about removing the Mid-Level Manager.

Chapter 4. Starting and Stopping Tivoli NetView

This chapter provides the necessary steps for starting and stopping the Tivoli NetView program and its daemons. The following topics are described:

Startup Behavior of the netview Shell Script

This section defines the behavior of the netview shell script, which starts the Tivoli NetView program.

If you are on the Tivoli NetView server and have root permissions, the netview shell script first executes the /etc/netnmrc shell script. The /etc/netnmrc shell script starts the SNMP agent, the nettl facility (the network logging and tracing facility), if they are not running, and the daemons registered in the ovsuf startup file. The daemons are started using the **ovstart** command.

Then, the netview shell script executes the **ovw** command, which starts the graphical user interface.

For Solaris Only

The snmpdx daemon and the mibiisa daemon must be running for the Tivoli NetView server to work correctly. These daemons are part of Sun SEA technology. To run the snmpdx agent, enter the following command:

```
/etc/init.d/init.snmpdx start
```

Customizing Startup

Some Tivoli NetView customers and application vendors want to set environment variables or execute scripts when the **netview** command is executed.

To customize the startup process, modify the /usr/OV/bin/netnmrc.applsetup shell script, /usr/OV/bin/netnmrc.pre shell script, or /usr/OV/bin/netnmrc.aux shell script, rather than the netview and netnmrc shell scripts. The applsetup, the netnmrc.pre, and netnmrc.aux shell scripts reside in the /usr/OV/bin directory, which is automatically backed up and migrated by the update installation or when you select the /usr/OV/bin.USER category during migration. This prevents the possible loss of startup configuration because the netview and netnmrc (/etc/init.d/netnmrc on Solaris or /etc/netnmrc on AIX) shell scripts are subject to modification with each service update or new version of the Tivoli NetView program. See "Appendix E. Files That Migrate" on page 95 for more information.

Users or application vendors who want to set environment variables or execute scripts when the **netview** command is executed should make these modifications in the applsetup script. Each command must run its script in the current process if the script set or changes environment variables passed to the user interface at startup.

Note: To avoid the loss of startup customization, modifications to the **netview** script must be moved to the /usr/OV/bin/applsetup script.

The netview shell script runs the /usr/OV/bin/applsetup script (if it exists), just prior to starting the graphical user interface. The applsetup script is run in the same process as the **netview** command, and thus enables the setting or changing of environment variables and other customized actions to be performed just as though the code had been edited into the netview shell script itself.

See the netview man page for more information about editing the applsetup script.

If you want to start processes that run independently of the graphical user interface and that require root access, make these modifications in the netnmrc.pre or netnmrc.aux shell script. The netnmrc shell script runs the /usr/OV/bin/netnmrc.pre shell script, if it exists, before starting the daemons, and runs the /usr/OV/bin/netnmrc.aux shell script, if it exists, after starting the daemons. Entries in the netnmrc.pre or netnmrc.aux shell script do not run in the current process.

Note: To avoid the loss of startup customization, modifications to /etc/init.d/netnmrc on Solaris or /etc/netnmrc on AIX must be moved to the netnmrc.pre and netnmrc.aux scripts.

Preparing to Start Tivoli NetView

The server installation process starts all the daemons registered in the ovsuf file and checks the status of the daemons. If you have root permissions, the SNMP agent and all registered daemons are started when you start the driver graphical user interfaceusing the **netview** command or the Tivoli desktop.

To check the statuses of the daemons from the server, use the **ovstatus** command or use the Server Setup application. To use the Server Setup application, see “Checking Daemon Status Using Server Setup”. To check the statuses of the daemons from the client, use the **nvstatus** command.

Start any daemons that are not running. If the required daemons are not running, the graphical user interfacewill not execute. Have the system administrator start the daemons for you if you do not have root authority.

For information about restarting the Tivoli NetView daemons, see “Restarting the Daemons” on page 23.

Checking Daemon Status Using Server Setup

The installation process starts all the daemons registered in the ovsuf file. However, before you start Tivoli NetView, you might want to check the statuses of the daemons and start them if necessary. You do not need root permissions to check the statuses of the daemons, but you must have root permissions to start them. If the required daemons are not running, the graphical user interface will not run.

To check the statuses of the Tivoli NetView daemons, follow these steps:

1. Enter **/usr/OV/bin/serversetup** from the command line to start the Tivoli NetView Server Setup application.
2. Select **Control –> Display Tivoli NetView status –> Display status of daemons**.

   All the Tivoli NetView daemons and their statuses are displayed. as shown in Figure 2 on page 16.

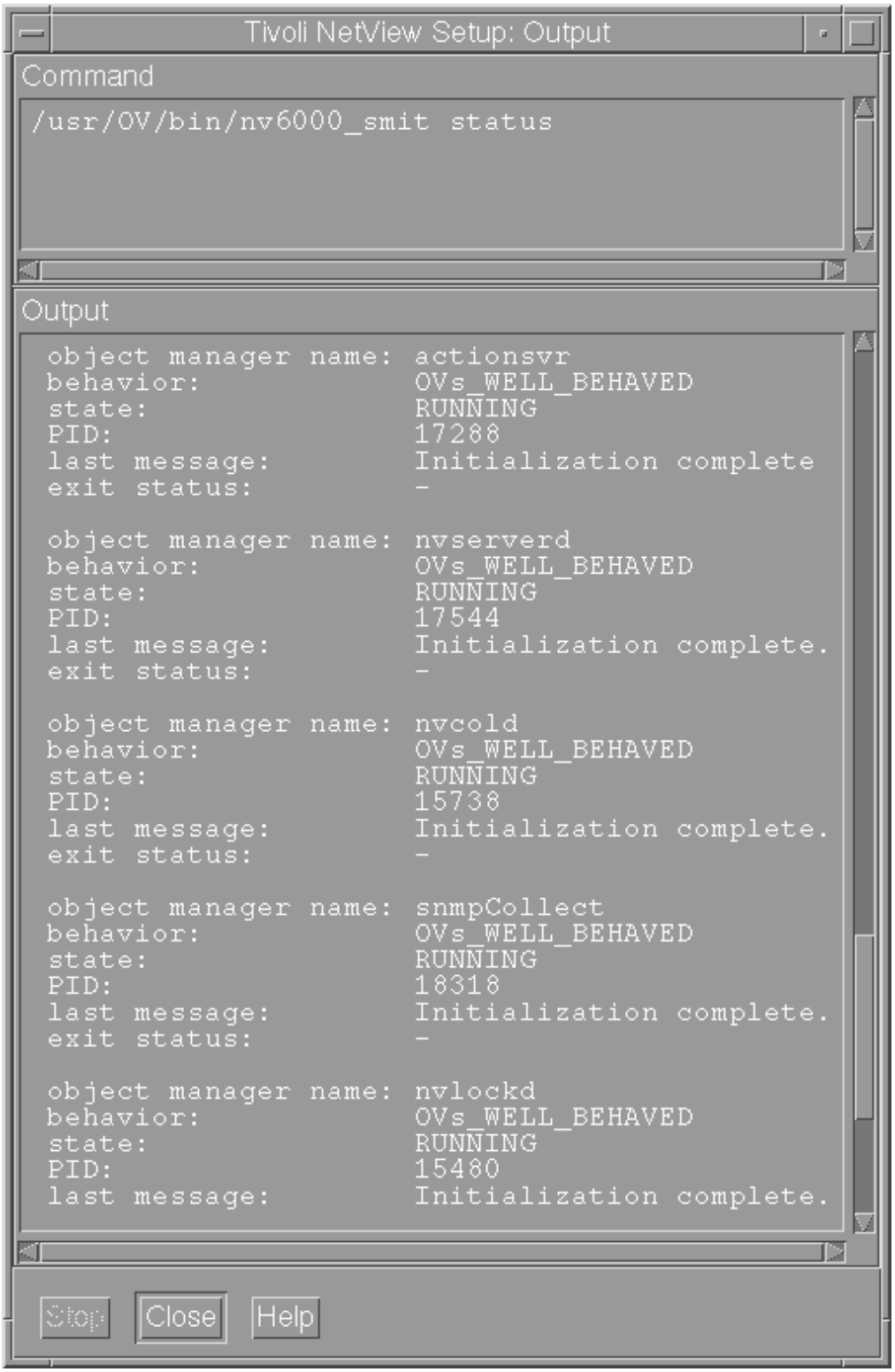

Figure 2. Display Status of Daemons Example

3. Click **OK** to close the window.

Starting Tivoli NetView

Start the Tivoli NetView program using the Tivoli desktop or the netview shell script. When you use the Tivoli desktop, the netview shell script is used to start the Tivoli NetView program.

Using the netview Shell Script

Use the netview shell script, which is executable whether you have root permissions or not, to start the Tivoli NetView program. See the netview man page for information about the command options. If the /usr/OV/bin directory is not in your PATH, either execute the **/usr/OV/bin/netview** command path or add the directory /usr/OV/bin to your PATH.

Starting Tivoli NetView Using the Tivoli Desktop

To start Tivoli NetView using the Tivoli desktop, follow these steps:

1. Enter **tivoli** on the command line to access the Tivoli desktop.
2. Click and hold down the right mouse button on the server or client icon to display pull-down menu.
3. Select **Control –> Start user interface**.

   The Start user interface dialog as shown in Figure 3 is displayed.

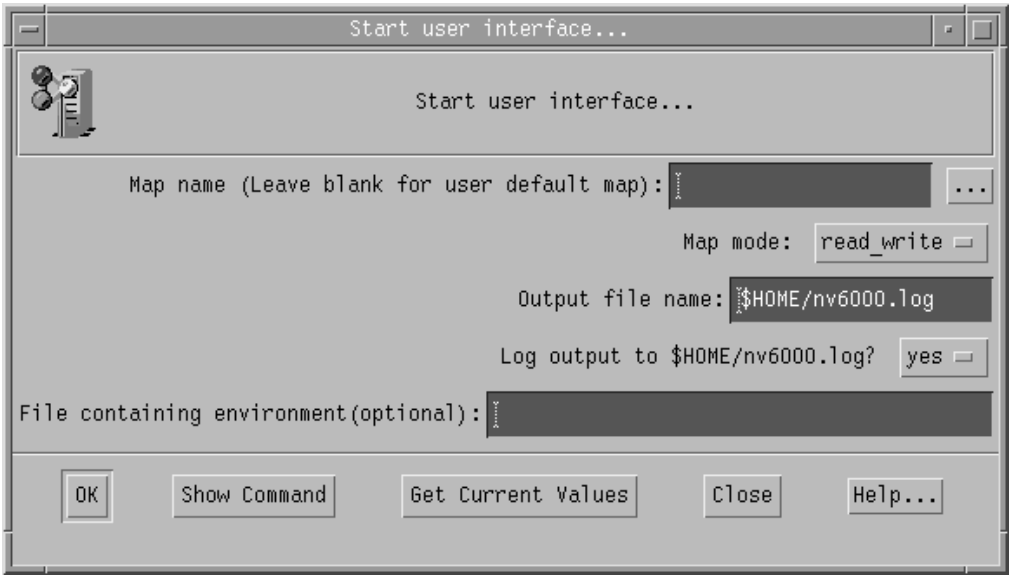

Figure 3. Start User Interface Dialog

   Click the buttons beside the entry fields to select the options you want.
4. Click **OK**.

   The Tivoli NetView program starts and the selected map is displayed.
5. To exit, select **Exit** from the Tivoli NetView **File** menu.
6. Click **OK**.

   The program closes.

Logging Output

In some cases, Tivoli NetView displays messages on the screen. Whether the Tivoli NetView program is started from the command line or through the Tivoli desktop,

these messages and output from integrated applications are also logged in the netview_$LOGNAME.log file (where LOGNAME is the UNIX login name of the user starting up NetView). All NetView log files default to usr/OV/log. You can specify an alternate location and name and edit the log file name in the log output field. The **-nl** option is useful if you have an application that writes realtime information to the stdout or stderr files, and you want to see the errors as they occur. For more information about some of the log file errors, refer to the *Tivoli NetView for UNIX Diagnosis Guide*.

You can change the option to log output through the Tivoli desktop or by starting the Tivoli NetView program using the **netview -nl** command. You might find this useful if you are running applications that produce a large amount of data. This prevents the log file from increasing and consuming system resources.

Generating the Map

When the daemons are first started, you can expect intense polling traffic, because the netmon daemon is working to discover objects on your network. The first time Tivoli NetView creates a map on a client, especially if the database is NFS mounted, the synchronization may take several minutes.

Generally, a client machine is smaller than a server machine, but the client has to "learn" all the map information that the server already knows. When the client brings up the graphical user interface, it synchronizes the information that it displays with the database information. The amount of time this takes varies according to the size of your network.

The graphical user interface creates and displays an interactive graphical map, which represents the logical topology of your network. For each map, an environment of interactive windows called submaps is created. A submap is a particular view of some part of the network that displays symbols that represent objects.

Defining the Network Management Region

The set of networks and nodes that the netmon daemon is monitoring define a management region. When the daemons are started for the first time, the default management region is the management system (the node on which the Tivoli NetView program is executing) and the networks to which it is directly attached. The map's initial management region displays networks or subnets, segments, and gateways. Unmanaged nodes are displayed in beige.

Customizing Your Map

To expand and customize your submap, use the submap menu's **Options..Manage Objects** operation and the Edit menu's options if you have a map with read-write authorization. After an unmanaged network is managed, the netmon daemon starts discovering nodes for that network. Any new networks that are discovered are discovered as unmanaged. The management region is preserved between invocations of the graphical user interface.

Customizing Map Layout

You may want to customize the IP Internet topology to reflect your network layout: geographically, hierarchically, or by some other criteria that is important to your company. You may create this level of customization in your maps using a location

file. A location file contains a list of networks, or ranges of networks, and the locations under which they should be displayed; locations can be nested to create a hierarchy.

For information about creating a location file, see “Using a Location File to Customize the Map Layout” on page 36.

Customizing Discovery

The initial management region can also be defined using a seed file. A seed file contains a list of host names, IP addresses, or a range of IP addresses. A seed file can be used to restrict network discovery or prioritize discovery so that the most important nodes are found first.

For information creating a seed file, see “Using a Seed File to Customize Discovery” on page 38.

Discovering IP Objects

The netmon daemon can discover only IP objects. Initially, the topology map will contain the following objects:

- IP networks, gateways, and routers on the Internet submap
- Segments, gateways, routers, hubs, and bridges on the Network submaps
- Hosts, gateways, routers, hubs, and bridges on the Segment submaps

Instead of computers, some IP nodes that are connectors, such as bridges and repeaters, or devices, appear on the map as hosts. However, connectors that support IP and SNMP appear on the map as the appropriate connector. Whether or not the nodes are appropriately represented on the map is dependent on the ability of the management system to map a sysObjectID to an OVW symbol type. For information about mapping symbols to nodes, see “Mapping Symbols to Nodes” on page 46.

If an IP node supports SNMP, automatic topology map generation detects its existence, physical address, IP address, type of device, system Object ID, MIB data, and other information. If an IP node does not support SNMP, automatic topology map generation can detect only its existence, physical address, and IP address.

Discovering Non-IP Objects

When the netmon daemon discovers an IP node, it forwards the SNMP traps to the trapd daemon. The noniptopod daemon registers with the trapd daemon for notification of events that indicate an IP address has been discovered.

The noniptopod daemon then issues an **snmpget** request for all of the object IDs (OIDs) listed in the oid_to_command file. If an agent responds to the **snmpget** request, the **start** command associated with the OID is executed using the IP address as a parameter. The application started by the **start** command is now responsible for discovering objects supported by the applications protocol and forwarding its topology data to the gtmd daemon. You must register the noniptopod and gtmd daemons before you can start them.

See “Registering and Unregistering the Daemons” on page 25 for information on how to register these daemons. For information about creating the non-IP protocol proprietary daemon, refer to the *Tivoli NetView for UNIX Programmer’s Guide*.

Displaying Nodes

When putting symbols on a map, the management system matches the sysObjectID to a symbol type to be used in the map. If the system can match a sysObjectID to a symbol type, the node will be shown with the appropriate symbol in the map; if not, the node will be represented as a generic symbol.

For more information about sysObjectID, see “Editing the oid_to_sym Registration File” on page 47.

A node appears as a gateway if its MIB value for IP forwarding is not zero, and the node has at least two interfaces. IP nodes with multiple interfaces on the same network may appear multiple times on segments. A node might also appear as a gateway if the gateway flag is set in the oid_to_type file.

For more information about the oid_to_type file, see “Editing the oid_to_type Registration File” on page 49.

If a device within your domain is not specified on your name server, or in the /etc/hosts file, its IP address will be displayed and not its device name.

Map Layout Dependencies

The map layout and usefulness is dependent on four things:

- Correct subnet masks
- Correct IP addressing
- Network design principles that aid isolation of network faults and traffic
- SNMP-based, MIB-I (RFC 1156), or MIB-II (RFC 1158) compliant agents throughout the network

The automatically generated topology map showing your networks or subnets and the gateways that connect them is based on your internetwork’s IP addressing scheme. It is crucial that IP network (subnet) masks are correct at least on the management system, all SNMP gateways, all SNMP routers, and all nodes listed in the seed file. Otherwise, the automatically generated topology map could contain incorrect networks with nodes from outside your administrative domain.

Network Design Principles

The following design principles can result in a more useful topology map layout:

- The logical breakdown of an internet topology into manageable networks or subnetworks through gateways and IP addressing. For example, you can subdivide a large network into several subnetworks, through IP subnetting, with gateways to route among the subnetworks.
- The physical breakdown of networks or subnetworks into manageable segments through repeaters, bridges, multiport repeaters, and gateways. For example, you can subdivide a large segment into several smaller segments connected through a multiport repeater.

If the automatically-generated topology map layout is not as useful as you would like, you can use the graphical network map’s editing operations to subdivide segments.

Accessing Online Help for the Graphical User Interface

When the graphical network map is displayed, you can use the online Help facility to find task-specific information.

Tivoli NetView uses Netscape Navigator or Netscape Communicator to display the online help. To access the list of help topics available, select **Help –> Help Topics** from the Tivoli NetView graphical user interface. You may also access the online help by clicking on **Help** in any Tivoli NetView dialog box.

Accessing Tivoli NetView Online Books

PDF versions are available in the /usr/OV/books/$LANG/pdf directory.

The Tivoli NetView books are available in PDF format. They are located in the **/usr/OV/books/$LANG/pdf** directory. The PDF versions of the books may be read and printed using the Adobe Acrobat Reader, which is available for download at the URL: **http://www.adobe.com**.

Using Server Setup to Configure and Manage a Tivoli NetView Server

The Server Setup application provides a menu-driven interface that lets you manage a Tivoli NetView server. It allows you to customize Tivoli NetView daemons, monitor the status of the Tivoli NetView daemons and applications, customize Tivoli NetView system files, diagnose Tivoli NetView problems, and maintain the Tivoli NetView databases.

To invoke the Server Setup application, use the following command:

```
/usr/OV/bin/serversetup
```

Or, you can add **/usr/OV/bin** to your PATH environment variable and simply type **serversetup**.

You can also invoke the Server Setup application by selecting **Administer –> Server Setup** from the Tivoli NetView graphical user interface.

On a Tivoli NetView client, the Server Setup application will prompt for a password on the server machine and will run remotely on the server machine. If no server has been set up for the client, this menu option will display the Client Setup menus.

Click on a menu item to expand it or select it. An open folder icon indicates an item that can be expanded; a file icon indicates an action to be executed. Actions that require input data will display an options dialog before executing.

Server Setup Context-Sensitive Help

The Server Setup application supports context-sensitive help on all menu items and option fields. To obtain context-sensitive help on a menu item or option field:

1. Select **Help**.
2. If you are in the main menu, select the **On Context** menu item. The cursor changes to a question mark (?). If you select **Help** in an options dialog, the cursor changes directly to a question mark.
3. Click the menu item or option field about which you want help.

Using Client Setup to Configure and Manage a Tivoli NetView Client

The Client Setup application provides a menu-driven interface that lets you manage a Tivoli NetView client. It allows you to configure the server for this client, start the Tivoli NetView graphical user interface, and deinstall the client.

To invoke the Client Setup application, use the following command:

```
/usr/OV/bin/clientsetup
```

You could also invoke the Client Setup application by adding **/usr/OV/bin** to your PATH environment variable and entering **clientsetup**.

Another way to invoke the Client Setup is by selecting **Administer –> Client Setup** from the Tivoli NetView graphical user interface.

Click a menu item to expand it or select it. An open folder icon indicates an item that can be expanded; a file icon indicates an action to be executed. Actions that require input data will display an options dialog before executing.

Client Setup Context Sensitive Help

The Client Setup application supports context sensitive help on all menu items and option fields. To obtain context-sensitive help on a menu item or option field

1. Select **Help**.
2. If you are in the main menu, select the **On Context** menu item. The cursor changes to a question mark (?). If you select **Help** in an Options dialog, the cursor changes directly to a question mark.
3. Click the menu item or option field on which you want help.

Restarting Automatic Map Generation

If a particular network on the map does not appear as it should, use the graphical user interface to edit the map. However, if the entire map does not accurately represent your network, you might want to restart automatic map generation. Root permissions are required to perform this task.

The following list describes examples of when you might want to restart automatic map generation:

- IP topology of your entire management domain has changed dramatically.
- Map or topology databases are corrupted.
- Manager station has moved to another network.

Steps for Restarting Map Generation

Because all topology databases are removed when you restart map generation, you might want to back up your existing databases before you restart automatic map generation. Root permissions are required.

To back up the databases, stop all the daemons and enter the following command:

tar -cvf /tmp/*filename* **/usr/OV/databases**

Where *filename* is the name of the file in which the data will be saved.

If you need to restore the data, enter the following command:

tar -xvf /tmp/*filename*

Where *filename* is the name of the file in which you saved the data you want to restore.

To restart automatic map generation, follow these steps:

1. Exit the Tivoli NetView graphical user interface if it is running.
2. Enter **/usr/OV/bin/serversetup** from the command line to start the Tivoli NetView Server Setup application.
3. Select **Control –> Restart automatic map generation**.

   A warning dialog is displayed.
4. Click **OK** to continue.

   All the daemons are stopped. All existing databases, except Agent Policy Manager definitions, SmartSet definitions, and master polling and discovery settings are deleted.

   The /usr/OV/log/trapd.log file, /usr/OV/log/ovevents.log file, and /usr/OV/log/ovevents.log.BAK file are removed.

   All the daemons are restarted.
5. Run the **netview** command.

   The Tivoli NetView program is restarted and a new map is generated. However, if automatic discovery was turned off before you restarted the daemons, you will have to restart automatic discovery separately.

Note: When you use the Server Setup application to restart map generation, all the daemons are stopped and restarted. When you use Server Setup application to clear map databases, all the daemons are stopped, but not restarted.

Restarting the Daemons

If the daemons stop while the manager system is running, restart them using the **ovstart** command or the Server Setup application

You must have root permissions to restart the daemons. You can restart all the Tivoli NetView daemons or select individual daemons to restart. You can also use the **/etc/netnmrc** shell script, which starts all the daemons including the snmpd daemon. If you are starting daemons individually, all prerequisite daemons are automatically started.

Restarting the Daemons from the Command Line

Restart the daemons using the **ovstart** command. To restart all the daemons, enter:

```
/usr/OV/bin/ovstart
```

The **ovstart** command starts the process management daemon, ovspmd, and all of the background daemons.

Note: The **ovstart** command does not start the SNMP agent if it is not running. If the SNMP agent is not running, enter one of the following commands to start it before starting the other daemons, or execute the **/etc/netnmrc** shell script.

For...	Enter...
AIX	startsrc -s snmpd
Solaris	/etc/init.d/init.snmpdx start

By using the process name parameter, you can start one or more particular processes. To restart one daemon, for example the netmon daemon, enter:

```
/usr/OV/bin/ovstart netmon
```

In general, the names that you use to start the processes are obvious, but there are a few exceptions:

If this is the common name...	Use this name with ovstart...
orsd	OVORS_M
ovelmd	ems_log_agent
ovesmd	ems_sieve_agent

Normally ovstart reports only if a process fails to start. The **-v** option requests "verbose" mode of operation, which produces information about what is occurring during the startup process. This option is useful for diagnosing problems. For example, to restart the netmon daemon with verbose mode, enter:

```
/usr/OV/bin/ovstart -v netmon
```

If a daemon requires other daemons to be running, the prerequisite daemons are started automatically.

Restarting the Daemons Using the Server Setup Application

To restart Tivoli NetView daemons using the Server Setup application, follow these steps:

1. Enter **/usr/OV/bin/serversetup** on the command line to start the Server Setup application.
2. Do one of the following:
   - Select **Control –> Restart all stopped daemons** to restart all the daemons.

     All the daemons are restarted. Go to Step 7 on page 25.
   - Select **Control –> Select daemons to stop or restart** to select individual daemons to restart. Go to Step 3.
3. Select one of the following:
   - Topology, discovery, and database daemons to stop or restart
   - Event and trap processing daemons to stop or restart
   - Host connection daemons to stop or restart (**AIX** only)
   - SmartSet and Agent Policy Manager daemons to stop or restart

   A dialog displays the names of the daemons.

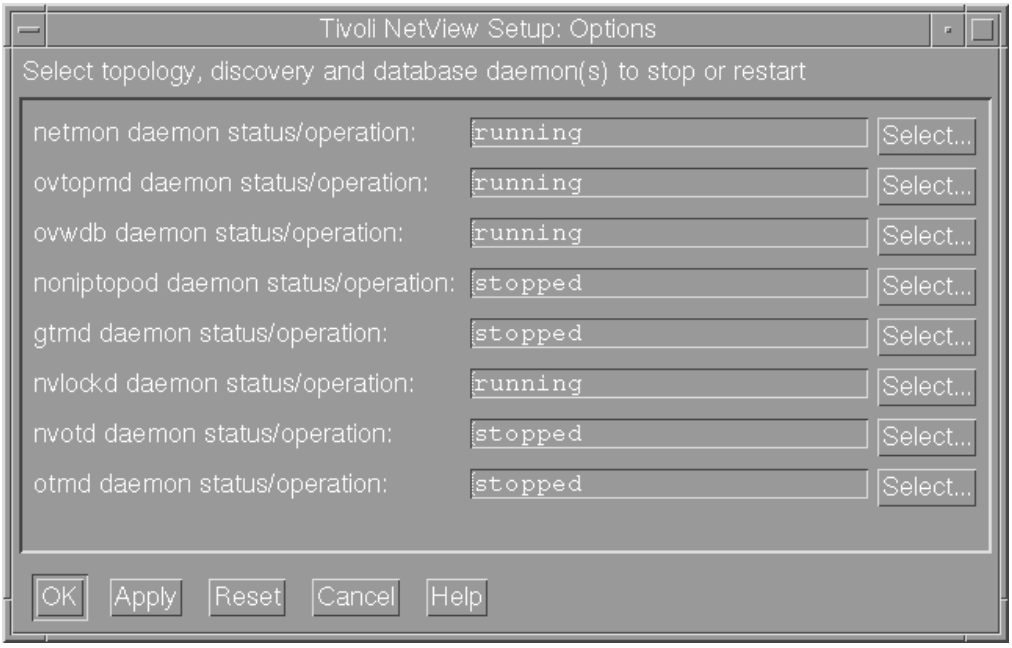

Figure 4. Daemon Stop or Restart Dialog

4. Select the button next to the daemon(s) you want to restart.
5. Select **restart**.
6. Click **OK**.

   All selected daemons are restarted.
7. Click **Close**.

   The dialog closes.

Registering and Unregistering the Daemons

The gtmd, noniptopod, otmd, C5d, tralertd, and spappld daemons are not automatically registered in the startup file. Therefore, these daemons are not started as part of the default startup process. These daemons must be registered before they can be started. After they are registered, they are started every time you execute the **netview** command as root and the **ovstart** command or you start the system, until you unregister the daemons.

You can use the Server Setup application or the **ovaddobj** command to register the daemons. The Server Setup application executes the **ovaddobj** command.

Registering and Starting the Daemons from the Command Line

Use the following commands to register and start a daemon:

/usr/OV/bin/ovaddobj /usr/OV/lrf/*daemon***.lrf**

/usr/OV/bin/ovstart *daemon*

Where *daemon* is the name of the daemon you are registering.

Registering and Starting the Daemons Using the Server Setup Application

To register the daemons using the Server Setup application, follow these steps:

1. Enter **/usr/OV/bin/serversetup** on the command line to start the Server Setup application.

2. Select **Configure –> Set options for daemons**.
3. Select one of the following:
   - Set options for topology, discovery, and database daemons
   - Set options for event and trap processing daemons
   - Set options for host connection daemons (**AIX** only)
   - Set options for Agent Policy Manager daemons
4. Select the daemon you want to register.

   The Set Options dialog for the selected daemon is displayed.

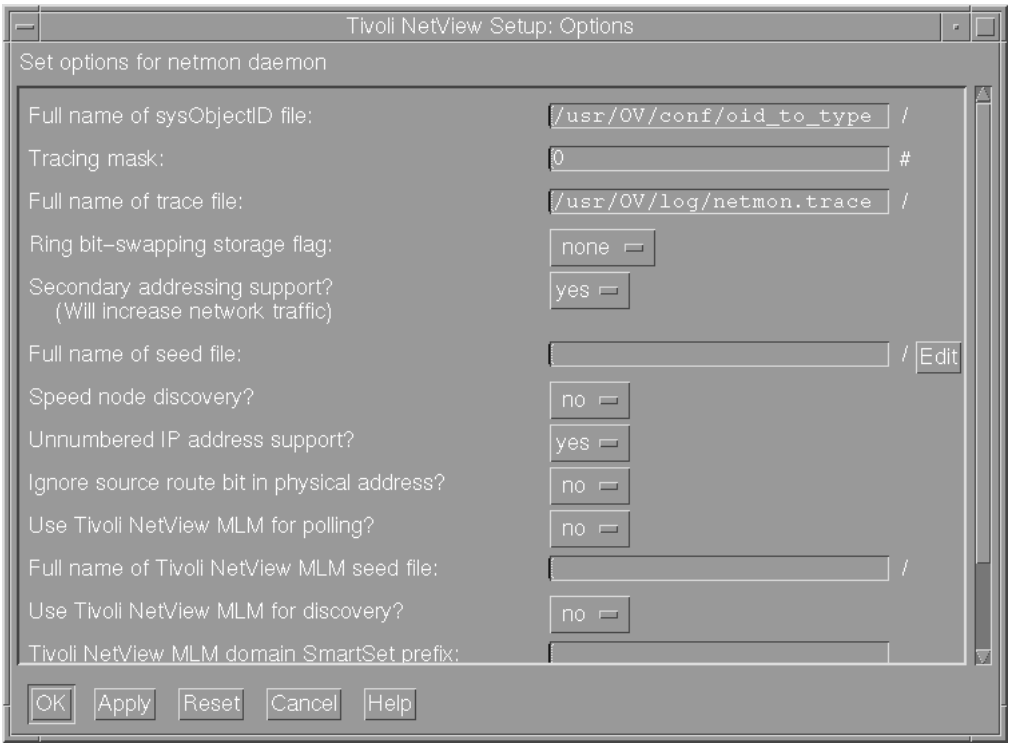

Figure 5. Daemon Dialog

5. Click **OK**. You do not need to change the defaults in the entry fields.

   The daemon is registered and added to the startup file.
6. Click **Close**.

   The dialog closes.

If you decide not to use these daemons and do not want them started, you must unregister the daemons.

See “Unregistering the Daemons” for those instructions.

Unregistering the Daemons

Using the Server Setup application, you can delete daemons from the startup file to prevent them from being started. You must have root permissions to perform this task.

Deleting unused daemons from the ovsuf file improves utilization of system resources. You can delete the following daemons from the ovsuf file:

- spappld (**AIX** only)
- tralertd (**AIX** only)

- gtmd
- noniptopod
- trapgend (**AIX** only)
- otmd

Unregistering the Daemons Using the Server Setup Application

To delete daemons from the ovsuf file, follow these steps:

1. Enter **/usr/OV/bin/serversetup** on the command line to start the Server Setup application.
2. Select **Configure –> Delete daemon from ovsuf startup file**.

   The delete daemon from ovsuf startup file dialog is displayed.
3. Click the button beside **Daemon** to delete.

   The selected daemon is displayed in the Daemon to delete field.
4. Click **OK**.

   The information is processed and a dialog appears with a message that the daemon is deleted.
5. Click **OK**.
6. Repeat Steps 3 and 4 for each daemon you are deleting.

   The selected daemons are deleted from the startup file, but the daemons are not stopped if they are running.
7. Click **Close**.

   The dialog closes.

If you decide to use the daemon you deleted, you must register the daemon in the ovsuf file before you start it.

For more information, see “Registering and Unregistering the Daemons” on page 25.

Stopping Tivoli NetView

To stop the Tivoli graphical user interface, select **Exit** from the **File** pull-down menu. Selecting Exit does not stop the daemons, only the graphical user interface.

If you want the Tivoli NetView program to continuously monitor and track changes to your network and the management system, always keep the daemons running, even if the graphical network topology map is not operational. If you are performing multiprotocol management, the gtmd and noniptopod daemons should be running also. If you are using the Agent Policy Manager application, the C5d daemon should be running. If you have a host connection (**AIX** only), tralertd and spappld should continue running.

Stopping the Daemons

Use the command line or the Server Setup application to stop the daemons. You must have root permissions to perform this task.

Stopping the Daemons Using the Command Line

To stop all the daemons using the command line, exit the graphical user interface and any other applications that use the daemons, then enter:

```
/usr/OV/bin/ovstop
```

All the daemons are stopped, except the nvsecd and ovspmd daemons. The nvsecd daemon must be running for the Tivoli NetView program to run (whether the security feature is on or off). The ovspmd daemon must be running if any of the other daemons are running (like nvsecd). If you stop nvsecd, all users are logged out (if security is turned on). Therefore, limit stopping nvsecd to workstation shutdown or problem resolution situations. You can stop the nvsecd and the ovspmd daemons individually.

To stop an individual daemon, such as the netmon daemon, enter:

```
/usr/OV/bin/ovstop netmon
```

Note: All daemons that depend on the specified daemon are also stopped.

To stop several daemons at the same time, list each daemon that you want to stop. For example, to stop the ovwdb, ovtopmd, and netmon daemons, enter:

```
/usr/OV/bin/ovstop ovwdb ovtopmd netmon
```

Note: The **ovstop** command does not stop the nettl facility. To stop the nettl facility, enter:

```
/usr/OV/bin/nettl -stop
```

Stopping the Daemons Using the Server Setup Application

To stop Tivoli NetView daemons, follow these steps:

1. Enter **/usr/OV/bin/serversetup** on the command line to start the Server Setup application.
2. Do one of the following:
   - To stop all the daemons, select **Control –> Stop all running daemons**.

     All daemons except ovspmd and nvsecd are stopped. Go to Step 8.
   - To stop individual daemons, select **Control –> Select daemons to stop or restart**. Go to Step 3.
3. Select one of the following:
   - Topology, discovery, and database daemons to stop or restart
   - Event and trap processing daemons to stop or restart
   - Host connection daemons to stop or restart (**AIX** only)
   - SmartSet and Agent Policy Manager daemons to stop or restart

   The selection dialog is displayed.
4. Click the button beside the daemon you want to stop.
5. Select **Stop**.
6. Repeat Steps 3 through 5 for each category of daemons you want to stop.
7. Click **OK**.

   All selected daemons are stopped.
8. Click **Close**.

   The dialog closes.

Chapter 5. Optional Configuration Tasks

This chapter describes additional ways you can configure the Tivoli NetView program. The following configuration options are described:

Changing Daemon Defaults

Changing the defaults for the daemons is not necessary for standard operation. However, you might want to configure the daemons to use values different from the defaults provided. For example, if you want netmon to start discovery using a seed file or if you have a device that supports secondary addressing, go to the netmon daemon dialog and change the defaults for those options.

You can change the defaults using the command line or the Server Setup application. This section describes the steps for using the Server Setup application. Refer to the *Tivoli NetView for UNIX Administrator's Reference* for information about using the command line to set daemon options. Using the Server Setup application prevents the possibility of creating errors in the LRF files. You must have root permissions to set options for the Tivoli NetView daemons.

The following changes occur when you change the daemon options using the Server Setup application:

- Updates the LRF files
- Updates the ovsuf startup files
- Stops and starts the daemons using the new values

From then on, when you start the system or execute the **netview** and **ovstart** commands, the new values are used.

You can change the defaults for the following daemons:

- Topology discovery and database daemons
  - netmon
  - ovtopmd
  - ovwdb
  - noniptopod
  - gtmd
  - otmd

- Event and trap processing daemons
  - pmd
  - orsd
  - trapd
  - trapgend (**AIX** only)
  - snmpCollect
  - ovelmd
  - ovactiond
- Host connection daemons (**AIX** only)
  - tralertd
  - spappld
- Agent Policy Manager daemon
  - C5d

Changing Daemon Defaults Using the Server Setup Application

To use the Server Setup application to change the defaults of the daemons:

1. Enter **/usr/OV/bin/serversetup** from the command line to access the Server Setup application.
2. Select **Configure –> Set options for daemons**. Then, select one of the following:
   - Set options for topology, discovery, and database daemons
   - Set options for event and trap processing daemons
   - Set options for host connection daemons (AIX only)
   - Set options for Agent Policy Manager daemons

   The selected menu is displayed.
3. Select a daemon.

   The Set options for daemon dialog is displayed similar to Figure 6 on page 31.

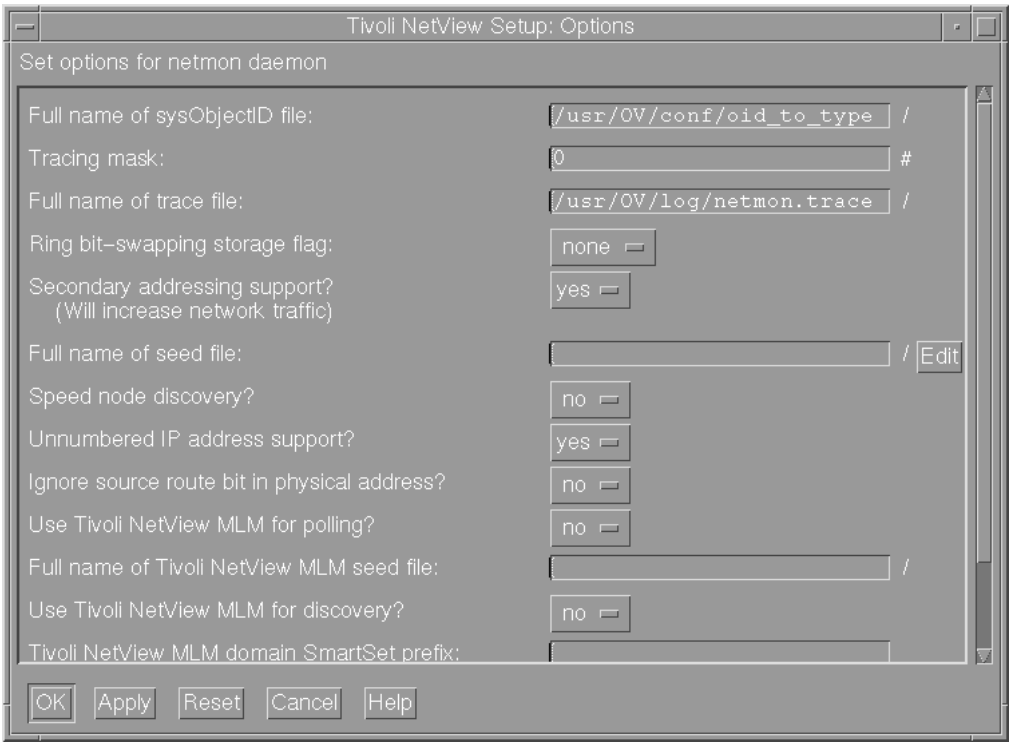

Figure 6. Daemon Dialog

4. Make the necessary changes to the defaults in the entry fields.
5. Click **OK**.

   The information is processed and the options are set.

   Note: The daemon may be automatically started, depending on whether all its dependencies are met.
6. Click **Close**.

   The dialog closes.

Understanding the Daemons, Options, and Defaults

This section lists the daemons, their options, and their defaults as they appear in the Server Setup application. Refer to the *Tivoli NetView for UNIX Administrator's Reference* or use the online help facility for more information about the options.

Topology Discovery and Database Daemons

Table 2 on page 32 lists all the topology discovery and database daemon options and defaults.

Table 2. Discovery and Database Daemon Options

Daemon	Option	Default
netmon	Full name of SysObjectID file	/usr/OV/conf/oid_to_type
	Tracing mask	0
	Full name of trace file	/usr/OV/log/netmon.trace
	Ring bit-swapping storage flag	none
	Secondary addressing support? (Will increase network traffic)	no
	Full name of seed file	
	Speed node discovery	no
	Unnumbered IP address support	no
	Ignore source route bit in physical address	no
	Use Tivoli NetView MLM for polling	no
	Full name of Tivoli NetView MLM seed file	
	Use Tivoli NetView MLM for discovery	no
	Tivoli NetView MLM domain SmartSet prefix	mlmDomain_
	Manage object added by loadhosts?	no
	Always use Alternate Community Names for polling?	no
ovtopmd	Do you want to use an SQL relational database	no
	Use port to receive requests over TCP/IP	yes
ovwdb	Number of objects to hold in cache	5000
	Use port to receive requests over TCP/IP	yes
noniptopod	Full name of configuration file	/usr/OV/conf/oid_to_command
gtmd	Full name of log file	/usr/OV/log/gtmd.log
	Full name of trace file	/usr/OV/log/gtmd.trace
	Seconds between storing data to database	900
	Buffers allocated for trap data before sending to xxmap	1000
otmd	Trace activity	no
	Maximum size of trace file (KB)	500

Event and Trap Processing Daemons

Table 3 on page 33 lists all the event and trap processing daemons, options, and defaults.

Table 3. Event and Trap Processing Daemon Options

Daemon	Option	Default
pmd	Allow the pmd to receive SNMP	traps
	Allow the pmd to receive Common Management Information Protocol (CMOT) request over TCP/IP	both (tcp and udp)
	Allow the pmd to receive CMOT events over TCP/IP	both (tcp and udp)
	Number of users to bind above the pmd	40
	Maximum time (minutes) to keep an association open without traffic	15
	Maximum time (minutes) to wait for association open without traffic	2
	Maximum number of network connections that pmd can support	1024
	Maximum time to wait for response to confirmed request to a local agent	4 (minutes)
	Maximum time to wait for response to confirmed request to a remote agent	6 (minutes)
	Use TMN CMIS/CMIP facilities	no
orsd	Time (minutes) to check if garbage collection needed	0
	Percent data in database before garbage collection needed	60
	Bytes of shared memory allocated to ORS cache (inbound)	100
	Bytes of shared memory allocated to ORS cache (outbound)	100
	Bytes of shared memory allocated to ORS cache (proxy)	100

Table 3. Event and Trap Processing Daemon Options (continued)

Daemon	Option	Default
trapd	Log events and traps	yes
	Full path name of log file	/usr/OV/log/trapd.log
	Full path name of trace file	/usr/OV/log/trapd.trace
	Hex dump all packets received/sent by trapd?	no
	Receive buffer size for TCP/UDP socket	9216
	Create socket connection for V1R1 applications?	no
	Maximum size of trapd.log file: (KB)	4096
	Full path name of trapd log maintenance script	N/A
	Forward specific traps as events to: (hosts)	N/A
	Forward specific traps to: (hosts)	N/A
	Forward ALL traps to: (hosts)	N/A
	Port used to receive SNMP traps over UDP	162
	Port used to receive SNMP traps over TCP	162
	Set Trapd connected applications queue size	2000
trapgend	Maximum size log file in kilobytes	100
	Maximum number of log files	5
	Full path name of log and trace file	/usr/OV/log/trapgend.log
	Throttle time (seconds) for trap generation	60
	Trace events	none
snmpCollect	Defer time when node down, minutes	60
	Full path name of trace file	/usr/OV/log/snmpCol.trace
	Maximum PDU objects	100
	Configuration check interval (minutes)	1440
	Maximum concurrent SNMP sessions	5
	Collect on unmanaged nodes	no
	Tracing is on at start up	no
	Verbose trace mode	no
	Polling interval for nvcold (in minutes)	60

Table 3. Event and Trap Processing Daemon Options (continued)

Daemon	Option	Default
ovelmd	Maximum size of ovevent.log file (KB)	128
ovactiond	Full path name of log file	/usr/OV/log/ovactiond.log
	Traces the execution of ovactiond?	no
	Make command output verbose?	no
	Maximum wait time for command to execute?	300

Host Connection Daemons

Table 4 lists all the options and defaults for the Host Connection daemons.

Table 4. Host Connection Daemon Options

Daemon	Option	Default
Tralertd	Tracing mask	0
	Full path name of trace file	/usr/OV/log/tralertd.trace
	Service point application name	
	Service point host name	N/A
	Using NETCENTER or GMFHS?	no
	Domain name	SNMP
	Standalone timeout	90
	Using tralertd database	no
	Clean tralertd database every (1-30) days	7
spappld	Service point host name	N/A
	Service point application name	
	Execute shell state	bsh (bourne)
	Execute shell path	/bin:/usr/bin:/usr/OV/bin
	Log service point transactions?	yes
	Full name of log file	/usr/OV/log/NV390.log
	Tracing mask	0
	Full name of trace file	/usr/OV/log/NV390.trace
	Using NETCENTER or GMFHS?	no

Agent Policy Manager Daemon

Table 5 on page 36 lists all the options and defaults for the Agent Policy Manager daemon, C5d.

Table 5. Agent Policy Manager Daemon Options

Daemon	Option	Default
C5d	Full path name of log file	/usr/OV/log/C5d.log
	Trace the execution of Agent Policy Manager?	no
	Full path name of trace file	/usr/OV/log/C5d.trace
	Number of minutes between daemon attempts	60
	Number of threshold events stored in history file	200

Using a Location File to Customize the Map Layout

Use the **/usr/OV/conf/location.conf** file to customize the IP Internet map. It determines how the networks are laid out on your map. It contains locations and the networks and gateways that should be placed within those locations. Only networks and gateways are placed under a location symbol. All other symbols, such as segments, are placed under the network to which they belong.

For examples of using the **location.conf** file, see "Location.conf File Examples" on page 37. For information about using **location.conf**, see "Location.conf File Usage Notes" on page 38.

After creating or modifying the **location.conf** file, either create a new map or regenerate the default map so that the **location.conf** file can take effect for networks already existing in your map. As new networks are discovered, they will be placed as specified in the **location.conf** file.

To create a new map, select **File –> New Map** from the NetView console.

To delete the current maps and restart map generation, select **Control –> Restart Automatic Map Generation** from the Server Setup application. You can access this application from the NetView Console (**Administer –> Server Setup**) or from the command line by running the **/usr/OV/bin/serversetup** command.

Refer to the sample **location.conf** file in the **/user/OV/conf** directory for the syntax of the file. Invalid entries are documented in the **/usr/OV/log/location.log** file and ignored.

Network Entries

Network entries have the following format:

```
Name AddressPattern LocationIcon <ParentName>
```

Where:

Name Name for the location symbol displayed on the map.

AddressPattern
Networks whose network addresses match this pattern will be placed under the location symbol. See the **/usr/OV/conf/location.conf** file for details on the syntax of this pattern.

LocationIcon
Type of location icon to be used. Possible values include: Site, Room, and

City. Refer to the **Help –> Legend** menu item from the Tivoli NetView console for a list of all possible values for the Location icon.

ParentName
Optionally, a previously defined parent location under which this location symbol should be placed. Using the parent location enables locations to be nested. Forward references are not valid.

Auto-placement of Routers

A router is placed in the lowest nested location that includes all of the networks to which the router is connected as follows:

- If all of the router's networks are inside one location, the router is displayed in this location connected to all the networks.
- If the router's networks are in more than one location, the router appears in the first location (up the parent chain), which includes all of the locations for these networks. The router is displayed in the parent location connected to one or more child locations.
- If the router's networks have no common location in the parent chain, or if one of the networks is displayed in the IP Internet submap, the router is displayed in the top level IP Internet submap.
- If the router is discovered as unmanaged, it is displayed in the top level IP Internet submap (its interfaces have not been discovered in this case).

Gateway/Router Entries

Gateway entries in file **location.conf** enable you to specify the exact placement of routers. Use gateway entries if auto-placement entries do not meet your requirements.

Gateway entries have the following format:

```
locationName address
```

Where:

locationName
Name of the location under which the gateway is placed (for example, `London` or `"London England"`

Address
The IP address of one of the gateway's interfaces (for example, `westford-gate.ma.dev.tivoli.com` or `146.84.242.221`).

Location.conf File Examples

Following are examples of location files. See "Location.conf File Usage Notes" on page 38 for more information about using the location.conf file.

Example 1

This location file creates a series of location symbols representing smaller Company ABC offices. All machines in London have addresses in the range 146.84.223.* to 146.84.226.* (specified as 146.84.223–226). These machines and their associated segments and subnets would be placed under a location symbol called *london* which would live at the IP Internet level of the map. Each location symbol uses the "Location:Site" icon. Router `router1` is placed under the `London` location.

```
scdev        146.84.211-214 Site
sanfrancisco 146.84.218     Site
switzerland  146.84.219     Site
london       146.84.223-226 Site
london       router1.london.com
```

Example 2: This location file creates a nested series of location symbols under a location called SmallOffices. Only the SmallOffices location symbol appears at the IP Internet level, the other location symbols are under SmallOffices. Note that the AddressPattern field for SmallOffices is 0. This implies that no networks should exist under this symbol, just other location symbols. There are two entries for the SuiteB location. Networks that match either of these two address patterns will be added under the SuiteB location symbol

```
SmallOffices 0              Site
SuiteA       146.84.210     Room     SmallOffices
SuiteB       146.84.218     Room     SmallOffices
SuiteB       146.84.223-226 Room     SmallOffices
SuiteC       146.84.227     Room     SmallOffices
```

Location.conf File Usage Notes

This section provides information about using the **location.conf** file.

- There is no option to use a different file. If the **/usr/OV/conf/location.conf** file is present, it is used.
- Syntax errors in the **location.conf** file are documented in the **/usr/OV/log/location.log** file.
- A parent location must be defined in the **location.conf** file prior to any child gateway entries (this is similar to the restriction on nested child location entries).
- Gateway entries take precedence over auto-placement entries.
- If conflicting gateway entries exist in **location.conf**, the first address that matches one of the gateway's interfaces is used.
- You can enclose **location.conf** file entries in quotation marks.
- If a **location.conf** file entry includes a blank space, then the entry must be enclosed in quotation marks.

Using a Seed File to Customize Discovery

A ***seed file*** is an ASCII file that helps you specify to the netmon discovery daemon which nodes in your network should and should not be discovered. A seed file can contain IP addresses, host names, groups of nodes, and optional comments. The devices listed in the seed file should support SNMP.

Non-SNMP Devices in a Seed File

Listing non-SNMP devices (or SNMP devices whose community names you do not know) in the seed file can negatively affect the discovery process. In general, netmon determines the device subnet based on the IP address and subnet mask of the last router in the path to the device. If netmon does not know the path to the device (for example, if the device is listed at the top of the seed file), netmon tries to get SNMP information from the device to determine the subnet mask. If the device is non-SNMP and netmon does not know the path, netmon looks at the class of the IP address and takes the corresponding subnet mask.

If there is a non-SNMP device in your seed file whose subnet mask does not match the class of the IP address, netmon does not know the path to the device. Therefore, you should not put non-SNMP devices in your seed file.

Editing the Seed File Using Server Setup

Use the Server Setup application to add, change, or delete entries in the netmon seed file. You must have root permissions to perform this task.

Complete the following steps to edit the seed file:

1. Enter **/usr/OV/bin/serversetup** from the command line to start the Server Setup application.
2. Select **Configure –> Set options for daemons –> Set options for topology, discovery, and database daemons –> Set options for netmon daemon**. The Set Options for netmon Daemon dialog is displayed.
3. Click **Edit**. The Network Monitor Seed File Editor is displayed.
4. Select one of the following tabs:
   - Select the **Discovery** tab to add, remove, or edit discovery entries containing hostnames, IP Addresses, ranges of IP Addresses, or SysObjectIDs.
     a. Select either **Discover by IP Address** or **Discover by OID** from the drop-down list
     b. Use Add, Remove, or Edit from the appropriate section. The first section is for nodes to be included in discovery, the second for nodes to exclude (negative entries).
   - Select the **Special Feature**s tab to add, remove, or edit SNMP Status or HSRP entries.
     a. Select either **SNMP Status Checking** or **HSRP** from the drop-down list.
     b. Use Add, Remove, and Edit.
   - Click **Save** to save your changes.
   - Select **Close** from the system menu (accessible by clicking the icon in the left corner of the title bar) to close the Network Monitor Seed File Editor.
   - Click **OK**.

     The netmon daemon is stopped and then restarted using the modified seed file.
   - Select **File –> Exit** from the menu bar to exit the Server Setup application.

Format of a Seed File

The format of the seed file is a list of host names, IP addresses, or groups of SNMP nodes within your administrative domain. The nodes in the seed file can be either a specific node or groups of nodes. A specific node can be an IP address or a host name. A group of nodes can be an IP address range, multiple host names, or nodes specified by object OID. You can specify a group of nodes using the following methods:

- Specify an IP address range, by using the hyphen (-) to indicate a range of numbers or an asterisk (*) to match any number. Here are two examples:

  ```
  9.67.179.70-79
  9.*.160.*
  ```

- Specify multiple host names by using an asterisk (*) to match zero to any number of characters or a question mark (?) to match any one character.

  The following group of nodes matches all company nodes starting with hdcp:

  ```
  hdcp*.company.com
  ```

  The following group of nodes matches router1.company.com and router2.company.com:

  ```
  router?.company.com
  ```

- Specify multiple nodes by OID. You can use an asterisk (*) at the end of an OID to match any OID beginning with the OID specified before the asterisk. Use the asterisk only at the end of an OID. If you do not use a pattern matching (wildcard) character in the OID, the OID must match exactly.

  The following group of nodes matches all Windows** 95 nodes:

  ```
  @oid 1.3.6.1.4.1.311.1.1.3.2  #Windows 95 nodes
  ```

  The following group of nodes matches all Cisco products beginning with an OID of 1.3.6.1.4.1.9:

  ```
  @oid 1.3.6.1.4.1.9.* # All Cisco products
  ```
- Specify nodes that you do not want netmon to discover by preceding specific nodes or groups of nodes with an exclamation mark (!). If a group of nodes is specified without the ! operator and an overlapping group of nodes is specified with the ! operator, the ! operator has the higher priority. Nodes in the overlapping range will not be discovered. If you have listed specific nodes in the seed file that also fall within a range of nodes specified with the ! operator, the specific nodes have the higher priority. The specific nodes will be discovered.
- Specify nodes that you want to status poll using SNMP instead of PING by preceding the nodes with a dollar sign ($). For these nodes, netmon will use SNMP queries for ifAdminStatus and ifOperStatus to determine the status.
- Specify HSRP virtual interfaces by preceding the IP Address with the % character.

The seed file must have one entry per line. A # character indicates the beginning of a comment; other characters from the # character to the end of the line are ignored.

Note: You choose the name and location of the seed file. The seed file name you choose cannot contain a colon. Seed file names are saved in the LRF file for netmon. Because the LRF file uses a colon as a separator, do not use a colon in the seed file name.

Seed File Format Examples

The following example shows the format of a seed file that expands the initial discovery process:

```
nodel.division.company.com
router4.division.company.com      #Gateways make the best seeds
9.67.1.5
```

The following example shows the format of a seed file that limits the discovery process:

```
router2.division.company.com
router3.division.company.com
9.67.179.70-79
9.67.179.200
9.*.160.*
```

Routers are included in this example to provide a path to the nodes within the IP address range 9.67.179.70-79 that are more than one router away from the management station.

How netmon Uses a Seed File

The following factors affect how netmon uses a seed file:

- Whether New Node Discovery is turned on or off.

- Whether there are specific IP addresses or host names specified in the seed file. Listing specific IP addresses or host names in the seed file tells netmon which nodes to discover.
- Whether there are groups of nodes specified in the seed file. Listing groups of nodes in the seed file tells netmon which nodes not to discover. The netmon daemon will not discover nodes that are not included in the groups of nodes, for example, nodes that are outside of an IP address range.
- Whether there are specific nodes or groups of nodes preceded by an exclamation mark (!) in the seed file. The netmon daemon will not discover these nodes.

Table 6 summarizes how discovery can be affected by the New Node Discovery setting and the contents of the seed file:

Table 6. Discovery Process Using a Seed File

New Node Disc Switch	Groups of Nodes	Specific IP Addresses and/or Names	Minimum discovered (*)	Maximum discovered
OFF	No	No	Tivoli NetView workstation	Tivoli NetView workstation
OFF	No	Yes	The nodes listed in the seed file	The nodes listed in the seed file
OFF	Yes	No	Tivoli NetView workstation	Tivoli NetView workstation
OFF	Yes	Yes	The nodes listed in the seed file	The nodes listed in the seed file
No	No	Tivoli NetView workstation	All nodes in your network	
ON	No	Yes	The nodes listed in the seed file	All nodes in your network
ON	Yes	No	Tivoli NetView workstation	All nodes that reside within the group of nodes
ON	Yes	Yes	The nodes listed in the seed file	All nodes that reside within the group of nodes
Note: Netmon discovers the node on which netmon resides, regardless of the seed file or the New Node Discovery settings.				

Discovering Specific Nodes

If you list specific IP addresses and host names in a seed file, netmon will discover only the nodes of those addresses and names. If you place a specific IP address in a seed file, the node that has that IP address will be discovered as long as the IP address responds to a ping.

When you are building your seed file, consider the following facts:

- Names of the nodes listed must be resolvable for netmon to start when processing the seed file. For example, if the node name `bogus` was listed in the seed file and was not resolvable in th network, netmon would not start. If that occurs, the **ovstatus** command indicates that a seed file problem has occurred and the /usr/OV/log/netmon.trace file contains the name of the failing node. The name of the unresolvable node should be removed from the seed file and netmon should be restarted.
- If New Node Discovery is turned off, netmon still tries to discover the nodes that have IP addresses listed in the seed file.

- If New Node Discovery is turned on, netmon first tries to discover all of the nodes that have IP addresses in the seed file, and then tries to discover additional nodes not listed in the seed file.
- If you cannot ping an IP address in the seed file, netmon cannot create a node for the IP address.
- If netmon discovers an IP address in the seed file in a network that has not been discovered, netmon creates and manages that network.
- If netmon discovers a node, it discovers all IP addresses on that node, regardless of whether those IP addresses are listed in the seed file.
- netmon always discovers the node on which netmon resides regardless of whether New Node Discovery is on, and regardless of whether that node is listed in the seed file.
- You should not list non-SNMP devices in a seed file. Doing so can negatively affect the discovery process.

  See “Non-SNMP Devices in a Seed File” on page 38 for more information.

Limiting Discovery

You can eliminate nodes from discovery in either of the following two ways:

- Listing one or more IP address ranges, multiple host names, or nodes specified by OID in a seed file tells netmon to ignore all IP addresses outside those groups of nodes. A group of nodes is very much like a filter.
- Listing specific nodes or groups of nodes preceded by an exclamation mark (!) tells netmon to ignore these nodes.

When you are building your seed file, consider the following facts and exceptions:

- Specifying groups of nodes does not guarantee that netmon will discover nodes within those groups; it guarantees that netmon will not discover nodes that are outside the groups. To ensure that netmon discovers a node within a group of nodes, see “Discovering Nodes in a Seed File Range”.
- netmon discovers the node on which netmon resides regardless of whether New Node Discovery is on, and regardless of whether that node is within the groups of nodes listed in the seed file.
- When netmon discovers a node, it discovers all IP addresses on that node, regardless of whether those IP addresses lie outside the groups of nodes in the seed file.
- If New Node Discovery is turned off, netmon ignores whether you specified groups of nodes in the seed file.
- netmon discovers individual IP addresses and host names in a seed file regardless of whether they lie outside the groups of nodes in the seed file or regardless of whether they are included in a group of nodes specified with the ! operator.
- If a group of nodes is specified without the ! operator and an overlapping group of nodes is specified with the ! operator, nodes in the overlapping group will not be discovered.
- When netmon discovers a node, the node will not be deleted as a result of changing your seed file (although the node might be deleted for other reasons). You cannot redefine IP address ranges in the seed file to remove existing nodes from your map.

Discovering Nodes in a Seed File Range

To ensure that netmon discovers a node within a seed file IP address range, the following criteria must be met:

- The network in which the node resides must be managed in the IP topology database.
- netmon must be instructed to discover an IP address in the node.

There are various ways to make sure that the network in which the node resides is managed in the IP topology database:

- Manually add the network symbol using the graphical user interface.
- Make sure that the gateway leading to that network is discovered and manually manage the network symbol on the map.
- Put the node's address in the seed file. If a node's IP address is included in the seed file, netmon will automatically create its network.
- Put the IP address of another node that is in the same network in the seed file to tell netmon to create its network.

When the network is managed in the IP topology database, netmon needs to "learn" the existence of the IP addresses of the nodes you want netmon to discover. In most cases, after netmon discovers the network, netmon can discover the nodes inside the network by checking various tables in various MIBs of nodes that it has already discovered. In some cases, netmon might have no way of learning of the existence of that node. For example, the node might generate too little network traffic. When this happens, and you want to guarantee that netmon discovers the node, provide these specific IP addresses in the seed file. See "Understanding Examples of Seed File Setup and Usage" for examples.

Understanding Examples of Seed File Setup and Usage

This section provides examples of using a seed file.

Telling netmon Where to Start Looking for Nodes without Limiting Discovery

If you want Tivoli NetView to use your seed file as a starting point and then discover the rest of the nodes, create a seed file that contains only a list of individual IP addresses and host names using the Network Monitor Seed File Editor. See "Editing the Seed File Using Server Setup" on page 39 for information on using the NetWork Monitor Seed File Editor. See "Discovering Specific Nodes" on page 41 for a list of considerations for this seed file.

When you use this procedure, netmon is not limited to discovering the nodes in the seed file. You can use this procedure to ensure that netmon discovers certain nodes, especially nodes that are more than one hop beyond the management station.

Limiting Discovery to the Nodes Individually Listed in the Seed File

If you want Tivoli NetView to discover only the nodes in the seed file, complete the following steps:

1. Enter **netview** to start the Tivoli NetView graphical user interface.
2. Select **Options –> Topology/Status Polling Intervals: IP** from the menu bar.
3. Turn Discover New Nodes **off**.
4. Click **OK**.
5. Select **File**, then **Exit**, from the menu bar.

6. Create your seed file (see “Editing the Seed File Using Server Setup” on page 39 for information on using the Network Monitor Seed File Editor). Refer to “Discovering Specific Nodes” on page 41 for a list of considerations for this seed file.
7. Clear the topology databases (see “Clearing Databases” on page 67).
8. Restart the daemons by either starting the NetView graphical user interface as the root user, or start only the daemons using the Server Setup application or the **ovstart** command.

Limiting Discovery to a Range of Nodes Using Seed File Wildcards

To limit discovery to the nodes within a certain group of nodes, complete the following steps:

1. Create your seed file using the Network Monitor Seed File Editor. See “Limiting Discovery” on page 42 for a list of considerations for this seed file.

   At least one of the lines in the seed file must contain a pattern matching (wildcard) character (either * or - for IP address ranges; * or ? for multiple host names) or multiple nodes specified by OID. See “Editing the Seed File Using Server Setup” on page 39 for information on using the Network Monitor Seed File Editor.
2. Clear the topology database. See “Clearing Databases” on page 67.
3. Restart the daemons using the Server Setup application or the **ovstart** command.

Again, specifying a group of nodes in a seed file does not guarantee that netmon can discover anything within the group. When you specify a group of nodes, you are telling Tivoli NetView not to discover anything outside the group. To ensure that nodes within the group are discovered, you should explicitly list a few specific IP addresses in the seed file that fall within the group of nodes. A group of nodes is like a filter, and specifying a group of nodes by itself does not help the netmon daemon discover nodes.

You can use a group of nodes in a seed file to limit discovery to a list of specific IP addresses without turning the new node discovery option off. To do this, create a seed file that contains a list of IP addresses and a group of nodes that you know has no IP addresses. For example, if you want to discover only IP addresses 1.1.1.1 and 1.1.1.2, and you know that you do not have any IP addresses in the IP address range 3.*.*.*, you can create a seed file as follows:

```
1.1.1.1
1.1.1.2
3.*.*.*
```

This seed file would limit the discovery to the two IP addresses and would not require that you turn off the new node discovery option.

Configuring netmon to Use an MLM Seed File

You can configure the netmon daemon to use a MLM seed file or any other kind of seed file. When you configure netmon to use a seed file that contains a list of MLM nodes, the MLM polls nodes in its own domain and reports status changes to the Tivoli NetView program. The MLM seed file makes sure that your MLM nodes are discovered quickly. Refer to the *Tivoli NetView for UNIX Mid-Level Manager User's Guide* for more information.

To configure the netmon daemon to use any seed file, you must have root permissions. Complete the following steps:

1. Enter **serversetup** on the command line to start the Server Setup application.
2. Select **Configure –> Set options for daemons –> Set options for topology, discovery, and database daemons –> Set options for netmon daemon**.

   The set options for netmon daemon dialog is displayed.
3. Scroll to **Use Tivoli NetView MLM for Polling** and click **Yes**.
4. Type the full path name of the MLM seed file in the **Full name of Tivoli NetView MLM seed file** field.
5. Click **OK**.

   The information is processed. The netmon daemon will use the specified MLM seed file the next time the netmon daemon is started.
6. Click **Close**.

   The dialog closes.

If you are using a seed file to limit the discovery process, restart map generation to remove nodes from the database and put only those nodes listed in the seed file into the database. When you restart map generation, all existing databases are removed. When you restart the Tivoli NetView program, the seed file is used to generate a new map.

See “Restarting Automatic Map Generation” on page 22 for information about how to restart map generation.

After initial map generation using a seed file, any expansion of the topology map takes precedence over the contents of the seed file. However, if the seed file contains nodes not already on the topology map, these nodes and their corresponding networks are added to the topology map when the netmon daemon is started. The seed file is used every time the netmon daemon is started.

Changing File Owner, Group, or Mode

You can use the Server Setup application to change file owners, file groups, and file modes for all Tivoli NetView topology database files. You must have root permissions to perform this task. These changes do not affect the owner, group, or mode for directories.

To change an owner, group, or mode, follow these steps:

1. Exit the Tivoli NetView graphical user interface, if it is running, by selecting **File..Exit** from the menu bar.

   The graphical user interface windows close.
2. Enter **serversetup** on the command line to start the Server Setup application.
3. Select **Configure –> Change Map(s) owner/group/mode**.

   The Change Map(s) owner/group/mode dialog is displayed.

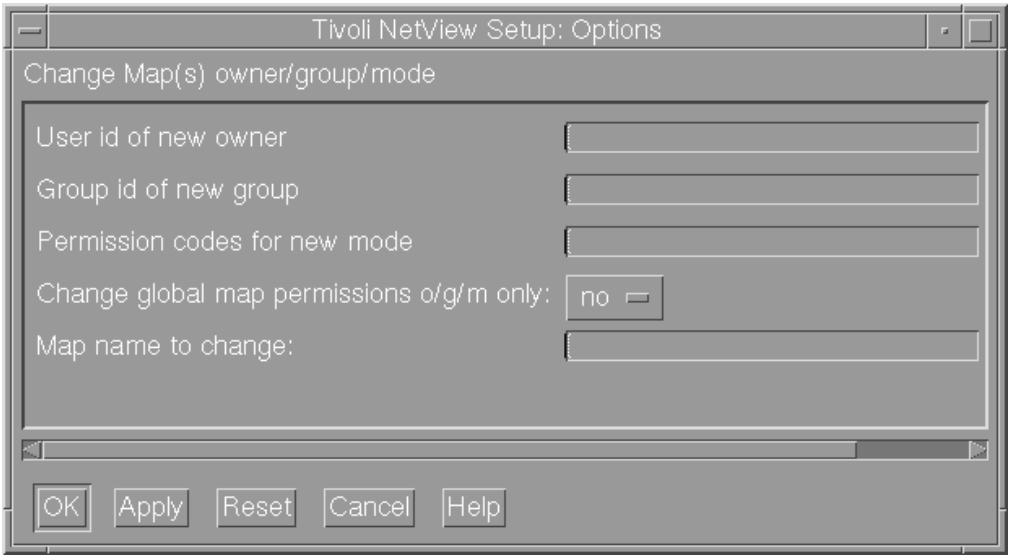

Figure 7. Change Map(s) Owner/Group/Mode Dialog

4. Type or select the entries for:
   - User ID of new owner
   - Group ID of new group
   - Permission codes for new mode: read (4), write (2), or execute (1). To specify a group of permissions, add together the appropriate octal numbers, as follows:
     3 = -wx (2 + 1)
     6 = rw- (4 + 2)
     7 = rwx (4 + 2 + 1)
     0 = --- (no permissions)
   - Change global map permissions owner/group/mode (o/g/m) only. Indicate **yes** or **no**.
   - Map name to change
   - List current permissions, owner, and group

   The information is displayed in the entry fields.
5. Click **OK**.
   The information is processed.
6. Click **Close**.
   The dialog closes.

Mapping Symbols to Nodes

The nodes in your network are represented in the graphical user interface as symbols. The Tivoli NetView program displays these symbols based on mappings from the sysObjectID to symbol types, vendors, SNMP agents, and IP topology attributes. This mapping is done using the following configuration files:

- oid_to_sym

  The IPMap and xxmap applications use the oid_to_sym configuration file. The /usr/OV/conf/C/oid_to_sym configuration file specifies a mapping from the sysObjectID of an agent to a default symbol class and subclass. The Tivoli

NetView program chooses an appropriate symbol type to represent nodes in the IP topology maps in the graphical user interface based on the value of sysObjectID.

- oid_to_type

  The netmon daemon uses the oid_to_type configuration file to map the sysObjectID of a node into the correct IP topology behavior and to the correct vendor and SNMP agent values. The /usr/OV/conf/oid_to_type configuration file specifies a mapping from the sysObjectID of an agent to the correct IP topology behavior and to the correct vendor and SNMP agent values. The oid_to_type file shipped with the Tivoli NetView product contains entries for specific network devices.

This section describes how you can edit these files so that the nodes in your network will be represented with an appropriate symbol in the graphical user interface.

Editing the oid_to_sym Registration File

The oid_to_sym file shipped with the Tivoli NetView program contains entries for various agents. You can edit these entries or add new ones.

Steps for Adding an Entry to the oid_to_sym File

Use the Server Setup application to add or change entries in the oid_to_sym file. Edit the oid_to_sym file using your text editor to delete entries from the file. You must have root permissions to perform this task.

To add an entry to the oid_to_sym file, follow these steps:

1. Enter **serversetup** on the command line to start the Server Setup application.
2. Select **Configure –> Configure object identification registration files –> Update oid_to_sym registration files**.

   The Update oid_to_sym registration file dialog is displayed as shown in Figure 8.

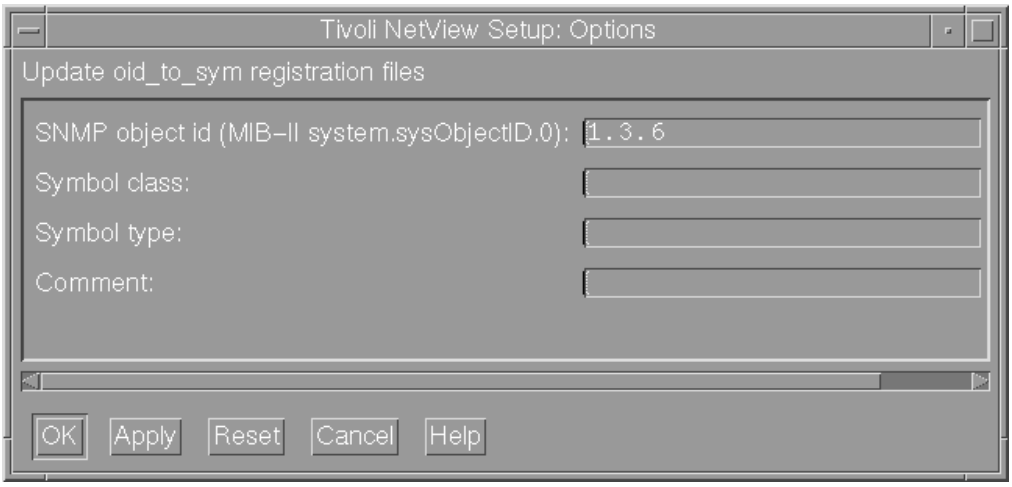

Figure 8. Update oid_to_sym Registration File Dialog

3. In the entry fields, type or select the required information.

   See “Field Definitions” on page 48 for information about the fields.
4. Click **OK**.

The information is processed, and the entry is added or changed.

5. Click **Close**.

   The dialog closes.

   Note: Existing symbols will not be changed. To change existing symbols, select **Edit..Change symbol type** operation from the Tivoli NetView graphical user interface.

Example of an oid_to_sym File

Each entry in the oid_to_sym file consists of three fields separated by a colon (:). The following example is from the default oid_to_sym file:

```
# IBM Network Enterprises
1.3.6.1.4.1.2.3.1.2.1.1.2:Computer:Workstation # IBM AIX workstation
1.3.6.1.4.1.2.2.1.2.2:Computer:PC              # IBM TCP/IP Agent on OS2
1.3.6.1.4.1.2.6.1:Connector:Bridge             # IBM 3172 LAN attachment
1.3.6.1.4.1.2.2.1.2.3:Computer:Main Frame      # IBM 3090 TCP/IP Agent
```

Field Definitions: The following list describes the fields used in the example of the oid_to_sym file:

- The first field is the value of the Internet-standard MIB-II system.sysObjectID reported by the device's SNMP agent. For example:

  `1.3.6.1.4.1.11.2.3.2.2.`
- The second field is the default symbol class. Valid symbol class entries are connector, computer, or device. The default value is **computer**.
- The third field is the default symbol subclass. Some valid symbol subclass entries for the computer class are workstation, mini, and PC.

The class and subclass fields together make up the OVW symbol type. The number symbol (#) indicates the beginning of a comment. Blank lines are ignored.

The values for symbol class and subclass must match one of the symbol types registered with the graphical user interface, or you can create new symbols. To see the currently defined symbols, see the /usr/OV/symbols/C registration files or **Help..Legend**.

To learn how to create new symbols, refer to the *Tivoli NetView for UNIX Programmer's Guide*.

Example of Changing a Symbol: If you want to change the symbol for an RS/6000 320 agent from the default symbol subclass Workstation to a minicomputer, edit the following line:

```
1.3.6.1.4.1.11.2.3.2.2:Computer:Workstation   # 320
```

to look like this:

```
1.3.6.1.4.1.11.2.3.2.2:Computer:Mini          # 320
```

When you add new symbol types you can add or change entries in the oid_to_sym file to take advantage of the new symbols. However, the symbol class and subclass must always match the list of symbols supported by the graphical user interface. If they do not match, ipmap may not be able to add the symbol to the IP topology map.

Editing the oid_to_type Registration File

You can edit the existing file and add or change lines as needed.

Making Changes to the oid_to_type File

Use the Server Setup application to add or change entries in the oid_to_type file. Edit the oid_to_type file using your text editor to delete entries from the file. You must have root permissions to perform this task.

To add or change an entry, follow these steps:

1. Enter **serversetup** on the command line to start the Server Setup application.
2. Select **Configure –> Configure object identification registration –> Update oid_to_type registration file**.

   The Update oid_to_type registration file dialog is displayed.

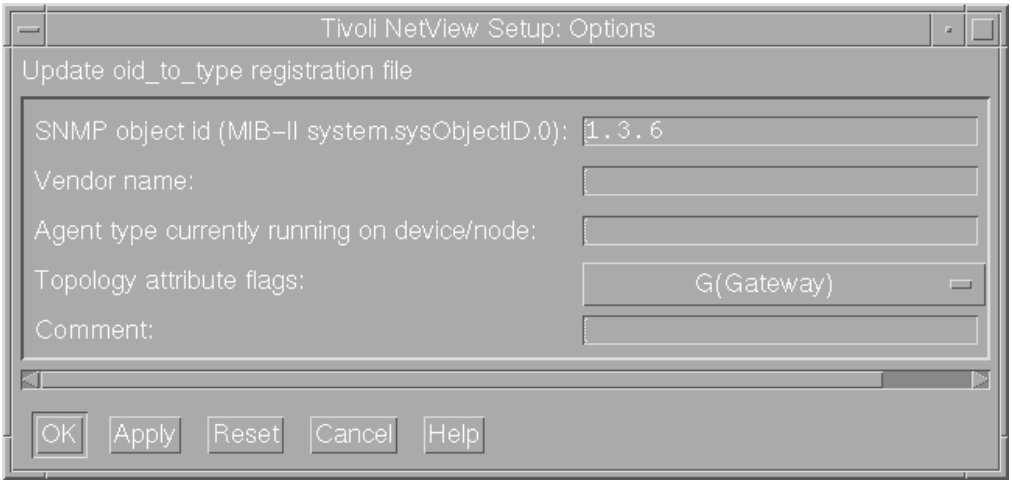

Figure 9. Update oid_to_type Registration File Dialog

3. Enter the required information, or click the button beside the entry fields to select from a list of choices.

   See “Field Definitions” for a description of these fields.
4. Select **OK**.

   The information is processed, and the entry is added or changed.
5. Select **Close**

   The dialog closes.

Example of an oid_to_type File

Each entry in the **oid_to_type** file consists of four fields separated by a colon (:). The following example shows sample entries in the oid_to_type file:

```
1.3.6.1.4.1.2.3.1.2.1.1.2:IBM:IBM RS/6000:H # SNMP agent for AIX 3.2
1.3.6.1.4.1.23.1.1.1:Novell:Novell Lantern
1.3.6.1.4.1.42.2.1.1:Sun:Sun Microsystems SunOS
1.3.6.1.4.1.36.1:DEC:DECstation
```

Field Definitions: The following list describes the fields used in an entry of the oid_to_type file, shown under “Example of an oid_to_type File”. These fields also apply to the entry fields used in the configuration utility.

- The first field is the value of the Internet-standard MIB-II system.sysObjectID reported by the device’s SNMP agent. For example: `1.3.6.1.4.1.23.1.1.1`.

- The second field is the vendor that manufactures the given device or node. The vendor name must match one of the enumerated values in the field registration file, /usr/OV/fields/C/ovw_fields, for the vendor field, such as IBM, Hewlett-Packard, Sun, or DEC. If the vendor name does not match, it is not set. You can add new values to the vendor field. See “Adding Values for Vendor and SNMP Agent Fields” on page 51 for more information.
- The third field is the agent type currently running on the given device or node. The agent name must match one of the enumerated values in the field registration file, /usr/OV/fields/C/snmp_fields, for the SNMPAgent field. The typical agent name includes the manufacturer of the agent software and the type of device on which the agent software runs, such as RS/6000and AIX Version 3 Release 2 SNMP agent.
- The fourth field is a set of flags controlling the topology attributes that should be applied to an object with this sysObjectID. The topology attributes field controls how the device will be treated by the IP-specific components of the system. These are only default attributes; if it can be determined that a device is or is not performing any of these roles, the attributes for that particular device will be set correctly. Specifically, any combination of flags may be applied to a sysObjectID.

Topology Attribute Flags: Table 7 describes the flags that can be used in the fourth field of the oid_to_type file:

Table 7. SNMP Topology Attributes

Flag	Meaning
G	Treat the device topologically as a gateway (router). A symbol for this object will appear in the Segment, Network, and Internet submaps, and the symbol can be used to connect networks.
B	Treat the object as a bridge or simple repeater. A symbol for objects with this sysObjectID will appear in the Segment and Network submaps, and the symbol can be used to connect segments.
H	Treat the object as a multiport repeater or hub. A symbol for objects with this sysObjectID will appear in the Segment and Network submaps, and the symbol can be used to connect segments. Also, this symbol can appear at the center (hub) of Star segments.
I	Ignore the node’s ability to support SNMP.
P	Poll these devices for status using SNMP instead of ICMP pings. Checks both ifAdminStatus and ifOperStatus.
S	Treat the device as if it supports secondary addresses but does not report them via SNMP in its ip.ipAddrTable. These devices have interfaces that are added, deleted, and then re-added by the netmon daemon in a regular pattern. The S option prevents the deletion of the secondary interfaces.
T	Report the node’s address as if it were a terminal server.
U	Treat the device as if it were unmanaged.
W	Assume that the device has a default Web management page at the URL “http://hostname.” The following fields will be created, if necessary, and set: **isHTTPSupported** True **isHTTPManaged** True **ManagementURL** http://hostname

Note: Special attributes such as “S” and “W” cannot be added using the Server Setup application. Edit the **oid_to_type** file using a text editor to add these special attributes.

Adding Values for Vendor and SNMP Agent Fields

To define additional values for the vendor and SNMP Agent fields, follow these steps:

1. Create a file with definition extensions.

   For example, if you work for the Cary Company and you want to add four entries to the SNMP Agent field, create a new file with the following information:

```
Field "SNMPAgent" {
        Type Enumeration;
        Flags    capability, general, locate;
        Enumeration "Unset",
            "Lou Router",
            "Calvin Hub",
            "Greg Bridge",
            "Ken Repeater";
}

Field "vendor" {
        Type Enumeration;
        Flags    capability, general, locate;
        Enumeration "Unset",
            "Cary Company";
}
```

2. Save your new file in the /usr/OV/fields/C directory.
3. Enter **ovw -fields** to add the new definitions to the object database.

The first four lines of the SNMP Agent field are identical to the definition lines in the snmp_fields file, and the first four lines of the vendor field are identical to the definition lines in the ovw_fields file. The values, except for "Unset", must be unique from those defined in the standard Tivoli NetView program.

For more information about the syntax, refer to the *Tivoli NetView for UNIX Programmer's Guide*.

The new definitions appear in the General Attributes selection when you select **Edit –> Modify/Describe** from the menu bar. The new definitions can be referenced in oid_to_type mappings described in "Editing the oid_to_type Registration File" on page 49.

Editing the oid_to_command Registration File

The /usr/OV/conf/oid_to_command registration file contains a list of OIDs and the associated commands provided by the proprietary protocol owners, for example, FDDI and Advanced Peer-to-Peer Networking (APPN). Each entry in the oid_to_command registration file includes the following:

- Object ID (required)
- Host name (required only if the host is remote)
- Start command (required)
- Stop command
- Comment

Use the Server Setup application to add an entry to the oid_to_command registration file. To change or delete entries, use your text editor to edit the file. You must have root permissions to perform this task.

Each entry in the oid_to_command registration file must conform to the following syntax:

```
OID [host name:]start command [options] |  [stop command] [options]
```

Adding Entries to the oid_to_command File

To add an entry to the oid_to_command registration file, follow these steps:

1. Enter **serversetup** at the command line to access the Server Setup application.
2. Select **Configure –> Configure object identification registration files –> Update oid_to_command registration file**.

   The Update oid_to_command registration dialog is displayed.

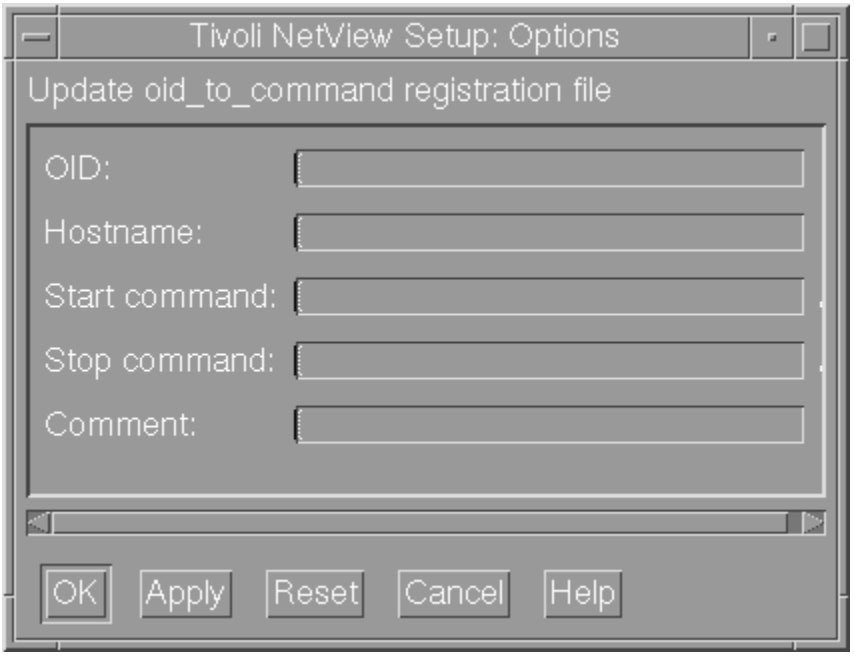

Figure 10. Update oid_to_command Registration Dialog

3. Type the required information in the entry fields.

   See “Field Definitions” for more information.
4. Click **OK**.

   The information is processed, and the new entry is added.
5. Click **Close**.

   The dialog closes.

Example of oid_to_command File

Following are examples of entries in the oid_to_command registration file:

```
1.0.8571.5.1 aixnmj01:/home/sam/my_daemon | /home/sam/3xtree/stop_daemon
1.0.56.78 /my_sub/bin/start_daemon | /my_other_sub/bin/stop_daemon
```

Note: The host name should be provided only if the host is remote. Do not add the host name if the host is local.

Field Definitions

The following list describes the fields used in an entry of the oid_to_command registration file. These fields also apply to the entry fields used in the configuration utility. See “Editing the oid_to_command Registration File” on page 51 for an example.

- The first field is the OID. This must be in dot-notation format (for example, 1.2.3.4.5.).

- The second field is the host name. A host name is necessary only when the host is remote. This is the name of the host where the proprietary protocol daemon resides. You must have authorization to execute processes on this remote host. The proprietary protocol daemon is expected to forward topology information to the gtmd daemon.

  For information about creating the protocol proprietary daemon, refer to the *Tivoli NetView for UNIX Programmer's Guide.*
- The third field is the full path name of the **start** command and options (if any). The **start** command is used to start the proprietary protocol daemon on the host.
- The fourth field is the full path name of the **stop** command and options (if any). The **stop** command stops the proprietary protocol daemon if the Tivoli NetView program should stop or shut down. This command will be used only if the proprietary protocol daemon was started using the start command as described above.
- The fifth field is a comment field.

Editing the oid_to_protocol Registration File

The /usr/OV/conf/oid_to_protocol registration file is a configuration file for open topology. The file contains a list of OIDs with text strings identifying the protocols used with those objects. You can add OIDs and their associated protocols for private use by editing the oid_to_protocol registration file. If you add entries to the oid_to_protocol file, you should add corresponding entries to the snmp_fields file. Refer to the *Tivoli NetView for UNIX Programmer's Guide* for more information. To add OIDs for public applications, contact the 10 Plus Association to request OIDs for new protocols. Otherwise, conflicts can result among vendors.

Example of an oid_to_protocol File

Each entry in the oid_to_protocol file consists of the OID and the text string that represents the protocol name. Following are examples of entries in the oid_to_protocol file:

```
# comment
"SNMP ifType"=1.3.6.1.2.1.2.2.1.3

${SNMP ifType}.1="Other"
${SNMP ifType}.2="Regular 1822"
${SNMP ifType}.3="HDH 1822"
${SNMP ifType}.4="DDN X.25"
${SNMP ifType}.5="RFC 877 X.25"
${SNMP ifType}.6="Ethernet CMSACD"
```

This could also be represented as:

```
# comment

1.2.6.1.2.1.2.2.1.3.1="Other"
1.2.6.1.2.1.2.2.1.3.2="Regular 1822"
1.2.6.1.2.1.2.2.1.3.3="HDH 1822"
1.2.6.1.2.1.2.2.1.3.4="DDN X.25"
1.2.6.1.2.1.2.2.1.3.5="RFC 877 X.25"
1.2.6.1.2.1.2.2.1.3.6="Ethernet CMSACD"
```

Redirecting X Window Display

You can choose the host system where you want your X Window to be displayed. To redirect your X Window display, follow these steps:

1. Ensure the management system has permission to display windows on the host you specified. If the management system does not have permission, use the **xhost** command on the host system. To do so, on the host system, enter:

   ```
   xhost + manager_hostname
   ```

   Where *manager_hostname* is the host name for the management system. Refer to the appropriate operating system documentation for more information about the **xhost** command.
2. On the management system, set the X Window DISPLAY variable by entering one of the following commands:

For...	Enter...
Bourne or Korn Shell	**export DISPLAY=***hostname***:0.0**
C Shell	**setenv DISPLAY** *hostname***:0.0**

   Where *hostname* is the host name of the system and display number to which you are redirecting the X Window display. The default display number is **0.0** (zero is the first display identified to the X-server).

Refer to the *Tivoli NetView for UNIX Diagnosis Guide* if you have problems with X Window.

Using a Relational Database for Data Storage

If the Tivoli NetView database component is installed, you can configure the Tivoli NetView program to use a relational database management system to store Tivoli NetView data. For example, you can configure Tivoli NetView to store IP topology data, trapd log data, and snmpCollect data in a relational database.

Tivoli NetView works with DB2/6000, Informix, Oracle, and Sybase. Refer to the *TME 10 Framework Reference* manual for the latest supported databases.

Refer to the *Tivoli NetView for UNIX Database Guide* for information about how to configure Tivoli NetView to store data in a relational database.

Configuring for Backup Manager

You can segment a large network by configuring a backup manager, which creates individualized spheres of control for each management station. Multiple Tivoli NetView programs can be used; each can be configured to cause minimal duplication of network management traffic.

Refer to the *Tivoli NetView for UNIX Administrator's Guide* for information about configuring a backup manager.

Configuring SNMP Values

Select the **Options –> SNMP Configuration** from the menu bar to configure SNMP values for SNMP communication. The SNMP Configuration menu item enables you to change the default values for the following items:

- Agent community names
- Proxies
- Timeout intervals and number of attempts
- netmon status polling intervals

You can also configure different default values for a specific node or a group of nodes. Configure a group of nodes by specifying an IP address global character (for example, 15.122.*.*) or a SmartSet. The IP address global character is useful when you want to configure different time-out values and number of retries for wide area networks (WANs).

For information about using the SNMP Configuration operation, refer to the online help or the *Tivoli NetView for UNIX Administrator's Guide*. For specific technical information, refer to the ovsnmp.conf man page.

Configuring Agent Community Names

A community name is a password that enables SNMP access to MIB values on an agent. The Tivoli NetView program works with agent community names in the following ways:

- By default, the manager's SNMP-based network management operations send the community name `public` in SNMP requests to agents.
- The manager's SNMP-based network management operations look up agent community names in the list shown in the SNMP Configuration dialog. The **Options –> SNMP Configuration...** operation allows network management operations to request MIB values from agents without requiring entry of a community name. By default, the only community name entered is `public`.

The /usr/OV/conf/ovsnmp.conf database contains node names or IP addresses with SNMP community names, timeout and retry intervals, and proxies. Changes you make using the **Options –> SNMP Configuration** operation are saved in the ovsnmp.conf database. If you are changing the community name on the manager workstation, you must change it in both of the following places:

- /usr/OV/conf/ovsnmp.conf database (the Tivoli NetView program must communicate with the SNMP agent)
- For **AIX**, /etc/snmpd.conf (the SNMP agent must use the new community name)

  For **Solaris**, /etc/snmp/conf/snmpd.conf and /etc/snmp/conf/snmpdx.acl

When to Configure Agent Community Names

You need to configure agent community names under the following conditions:

- If your SNMP agents have a community name other than public, use the **Options –> SNMP Configuration** operation to configure the management system to use the proper community name for the agents.
- If you want to set MIB values on an agent, you may also need to configure the SNMP agent to respond to SNMP SetRequests. Many SNMP agents do not support SetRequests, but the ones that do generally require you to enter a community name. How the community name is implemented and used depends on the agent. For information about an agent's community name, see the documentation provided by the vendor of the agent.

  For information about the SNMP agent, see the appropriate operating system documentation.

  If you plan to run APM, set up community names so that Tivoli NetView can do SNMP **Set** and **Get** requests to all your Tivoli NetView MLM hosts.

Using Alternate Community Names

You can add a list of up to six SNMP community names, used by various devices in your network, to the file **/usr/OV/conf/communityNames.conf**. When a netmon query to a device fails, netmon will try each of the community names in this file, in

an attempt to find the community name that works. An entry is then created for that node name/community name combination in the ovsnmp.conf database, so it will be used in future queries.

Normally, netmon will use the community name from the SNMP configuration. If a name has not been configured for a device, it will use *public*, the default. If this fails, netmon will use the list in communityNames.conf, but only under the following circumstances:

1. On initial discovery for all nodes.
2. After SNMP timeouts for routers, netmon will try names from the list. Many routers are configured so multiple interface IP addresses resolve to the same name. This capability enables Tivoli NetView to successfully use any of the IP addresses for SNMP queries.
3. When a Demand Poll fails, it will use the list.
4. A netmon configuration option enables netmon to use the alternate community name list and update its internal cache during its configuration polling if necessary.

   Use this option judiciously because it slows down polling noticeably due to timeouts for non-SNMP nodes. Ideally, you would want to use this option when a large number of community strings have been changed. It is intended to be used during a full configuration cycle (which is usually 24 hours) and then manually turned off.

   To enable this configuration option, select **Configure –> Set options for daemons... –> Set options for topology, discovery, and database daemons –> Set options for netmon daemon** in the Server Setup application. Turn on the **Always Use Alternate Community Names for all polling** field in the resulting dialog. You can access this application from the Tivoli NetView console (**Administer –> Server Setup**) or from the command line by running the **/usr/OV/bin/serversetup** command.

Authentication Failure

An authentication failure results when the community name, sent by a manager system to an agent, is not valid. When an agent receives a community name that is not valid, it can send an authentication failure trap to the Tivoli NetView program, which logs authentication failure traps in its event log, /usr/OV/log/ovevent.log.

Configuring a Proxy Agent

You can use a proxy agent to provide SNMP access to nodes that do not support SNMP. When you configure a proxy, the proxy agent receives the SNMP request and forwards it to the requested node using a non-SNMP protocol. How the proxy gets information from the target node depends on the target.

See the *Tivoli NetView for UNIX Programmer's Guide* for information about configuring a proxy agent.

Example of Using a Proxy: If you want to get information about a LAN Manager/X client, which is a PC node, the information does not come directly from the PC node, because the PC does not support SNMP. However, the LAN Manager/X server supports SNMP and the server can communicate with the PC node. In this example, you can configure the LAN Manager/X server to act as a proxy for the target PC node. All requests to the target PC node are really sent to the server.

Configuring netmon Polling Intervals

Select **Options –> SNMP Configuration** to change netmon polling intervals, such as status polling interval, fixed polling interval, and configuration polling interval in addition to other configuration parameters. The community name is also used by netmon, and the timeout interval is used as an initial estimate.

To set polling and discovery on or off, select **Options –> Topology/Status Polling Intervals: IP...** from the menu bar.

Configuring APM

The APM function is not automatically configured to run. To use APM, configure and start the APM daemon through the Server Setup application. The APM daemon is named C5d.

To configure the APM daemon, follow these steps:

1. Enter **serversetup** on the command line to access the Server Setup application.
2. Select **Configure –> Set options for daemons –> Set options for Agent Policy Manager daemon**.

   The **Set options for Agent Policy Manager daemon** menu is displayed.
3. Make the appropriate changes in the following fields:
   - Full path name of log file
   - Trace the execution of Agent Policy Manager?
   - Full path name of trace file
   - Number of minutes between daemon attempts
   - Number of threshold events stored in history file

   See the online help for more information about these fields.
4. Click **OK**.

   The information is processed.
5. Click **Close**.

   The dialog closes.

After configuring the C5d daemon, the daemon will be started automatically when you start Tivoli NetView.

Forwarding Events to the Tivoli Enterprise Console

You can configure Tivoli NetView to forward events to the Tivoli Enterprise Console event server. Using the Server Setup application, supply the host name on which the Tivoli Enterprise Console resides and a Tivoli NetView event correlation rule. See the *TME 10 Enterprise Console Rule Builder's Guide* for detailed information on rules. Then customize the Tivoli Enterprise Console event server to understand the format of the events coming from Tivoli NetView.

Configuring Tivoli NetView to Forward Events

To configure Tivoli NetView to forward events to the Tivoli Enterprise Console, follow these steps :

1. Enter **serversetup** on the command line to access the Server Setup application.
2. Select **Configure –> Configure event forwarding to T/EC**.

   The **Configuring event forwarding to T/EC** dialog is displayed.

3. Make the appropriate changes in the following fields:
   - Forward events to Tivoli event server?
   - Tivoli event server host name
   - NetView rule name

   See the online help for more information about these fields.
4. Click **OK**.
   The information is processed.

The configuration information is stored in the **/usr/OV/conf/tecint.conf** file. If you prefer, you can edit this file directly instead of using the Server Setup application.

The *NetView rule name* is an event correlation ruleset created with the Tivoli NetView Ruleset Editor. The following ruleset samples are provided with the Tivoli NetView kit:

Default.rs	No events are forwarded.
Forwardall.rs	All events are forwarded.
sampcorrIuId.rs	Forwards an interface down trap if a node up trap is not received for the same device within 10 minutes.
sampcorrNdNu.rs	Forwards a node down trap if a node up trap is not received for the same device within 10 minutes.

You may select one of these rulesets or create your own using the Tivoli NetView Ruleset Editor. If you select the Forwardall.rs ruleset, it is strongly recommended that you set up a filter inside the Tivoli Enterprise Console to reduce the number of events received by the console.

Only events that are true for the specified event correlation rule are forwarded to the Tivoli Enterprise Console event server. For information about the format of the forwarded events, refer to the *Tivoli NetView for UNIX Administrator's Guide*.

Customizing the Tivoli Enterprise Console Event Server

To customize the Tivoli Enterprise Console event server to understand the format of events coming from Tivoli NetView, perform the following steps on the event server:

1. Copy the following files to a directory on the host running the Enterprise Console:
   usr/OV/conf/nvserverd.rls
   usr/OV/conf/nvserverd.baroc
2. Create a rulebase for your environment. If a rulebase is already created, go to Step 4 on page 59. To create a rulebase, enter the following command:

   ```
   wcrtrb -d directory My_rb
   ```

   Where *directory* specifies the directory in which you copied the nvserverd.rls file and *My_rb* is the name of your rulebase.
3. Copy the contents of the Default rulebase into your newly created rulebase. Enter the following command:

   ```
   wcprb -cr -f Default My_rb
   ```

   Where *My_rb* is the name of your rule base.

4. Import the event classes. Change to the directory containing the nvserverd.rls and nvserverd.baroc files. Then enter the following command:

   wimprbclass nvserverd.baroc *My_rb.*

   Where *My_rb* is the name of your rulebase.
5. Import the rules.

   Enter the following command:

   wimprbrules nvserverd.rls *My_rb*

   Where *My_rb* is the name of your rulebase.
6. Compile the rulebase by running the following command:

   wcomprules *My_rb*

   Where *My_rb* is the name of your rulebase.
7. Stop the event server. Enter the following command:

   wstopesvr
8. Load the rulebase. Enter the following command:

   wloadrb *My_rb*

   Where *My_rb* is the name of your rulebase.
9. Restart the event server. Enter the following command:

   wstartesvr

If events are not forwarded to the console after you perform all the configuration tasks, enter the following command to see the raw data being received:

wtdumprl

Refer to the *Tivoli Enterprise Console User's Guide, Volume II* for more information about the commands used in this procedure.

Chapter 6. Maintaining Tivoli NetView

To optimize the performance of the Tivoli NetView program, you might need to perform various maintenance tasks. The following topics describe those tasks:

Maintaining Daemon and Process Logs

Maintain the daemon and process logs to make sure they do not grow too large and use up available file system space. A large log file can also adversely affect the performance of the Events subsystem. To prevent these problems, use one of the following methods:

- Periodically check the size of the log files and clear the contents as necessary.

  See "Clearing Log and Trace Files Using the Server Setup Application" for steps on using the Server Setup application to clear log and trace files.
- Create crontab entries to automatically clear log and trace files.

  See "Running Commands at Preset Times" on page 63 for steps on using the Server Setup application to set a crontab entry.
- Configure the trapd daemon to automatically archive trapd log data.

  See "Maintaining the trapd.log File" for steps on configuring the trapd daemon.
- Check disk space using the Systems Performance Monitor (shpmon) application.

  Refer to the *Tivoli NetView for UNIX Administrator's Guide* for instructions.

Clearing Log and Trace Files Using the Server Setup Application

To clear the contents of the log and trace files, follow these steps:

1. Enter **serversetup** on the command line to access the Server Setup application.
2. Select **Maintain –> Clear log, trace, or collector files**.

   The Clear log, trace, or collector files dialog is displayed.
3. Type the log or trace file you want to clear, or click the button beside the field for a list of choices.

   The selected file is displayed in the Log, trace, or collector file name field.
4. Click **OK**.

   The information is processed, and the selected file is cleared.
5. Click **Close**.

   The dialog closes.

Maintaining the trapd.log File

By default, the trapd daemon automatically clears the trapd.log file and moves the data to the trapd.log.old file when the trapd.log file reaches a specified size, 4096 KB by default. You can gain additional control over the trapd.log data by configuring the trapd daemon to run your own script or the trapd.log_Maint script when the trapd.log file reaches the specified size. Root permissions are required to perform this task.

The trapd.log_Maint script does the following processing of the data in the trapd.log.old file, depending on the parameters you set for the trapd.log_Maint script:

- Transfers the data to a relational database.

  Refer to the *Tivoli NetView for UNIX Database Guide* for information about transferring data to a relational database.
- Archives the data in the specified directory.

  The data is archived in a file that includes a Julian date and time stamp in the file name to indicate when the data was archived. For example, the file name trapd.log.94215153001 indicates that this file was archived on August 3, 1994 at 3:30:01 p.m.
- Discards archived data that is older than the specified maximum age.
- Verifies that the maximum amount of disk space used to store archived trapd.log data has not been exceeded. When the specified limit is reached, the oldest trapd.log data is discarded.

  Note: The archive maintenance actions do not affect trapd.log data stored in a relational database.

To maintain the trapd.log file by configuring the trapd daemon, follow these steps:

1. Enter **serversetup** on the command line to access the Server Setup application.
2. Select **Configure –> Set options for daemons –> Set options for event and trap processing daemons –> Set options for trapd daemon**.

   The Set options for trapd daemon dialog as shown in Figure 11 is displayed.

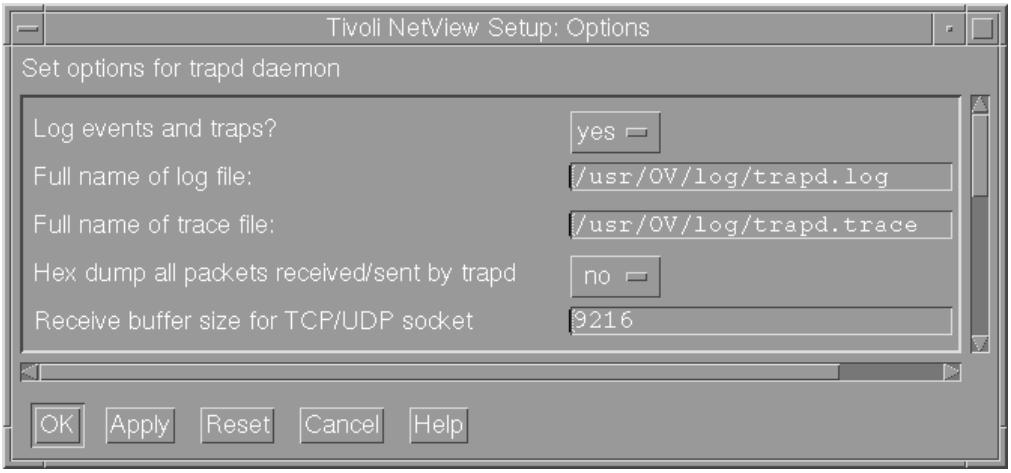

Figure 11. Set Options for trapd Daemon Dialog

3. Make the necessary changes to the entry fields:
   - Maximum size of trapd.log file
   - Full name of trapd log maintenance script

     Enter the full path name of any script you want to use, or click the button beside the field and select the trapd.log_Maint script from the list of choices.
4. Click **OK.**
   - If you did not select the trapd.log_Maint script, the trapd daemon is configured as specified. Skip to Step 6 on page 63.

- If you selected the trapd.log_Maint script, make the necessary changes to the trapd.log_Maint parameters that are displayed:
  - Directory for storage of archived trapd.log files
  - Maximum age of any archived trapd.log file
  - Maximum total size of all archived trapd.log files
  - Migrate data to SQL database

  Refer to the online help for additional information about the entry fields.

5. Click **OK**.
6. Click **Close**.

   The dialog closes.

Running Commands at Preset Times

If you have root permissions, you can use the Server Setup application to add crontab entries to run shell commands at preset dates and times. Adding crontab entries can help you organize and schedule various routine tasks.

Refer to the appropriate operating system documentation for information about the **crontab** command and job scheduling.

The Tivoli NetView program provides several shell scripts for routine maintenance. Using the Server Setup application, you can add a crontab entry for the following shell scripts:

netmon.trace _Maint
: Clears the netmon.trace file while keeping the last two versions of the file. The netmon.trace file is moved to /usr/OV/log/netmon.trace.BAK1 and the netmon.trace.BAK1 file is moved to /usr/OV/log/netmon.trace.BAK2.

snmpCol.trace_Maint
: Clears the snmpCol.trace file while keeping the last two versions of the file. The snmpCol.trace file is moved to /usr/OV/log/snmpCol.trace.BAK1 and the snmpCol.trace.BAK1 file is moved to /usr/OV/log/snmpCol.trace.BAK2.

trapd.trace_Maint
: Clears the trapd.trace file while keeping the last two versions of the file. The trapd.trace file is moved to /usr/OV/log/trapd.trace.BAK1 and the trapd.trace.BAK1 file is moved to /usr/OV/log/trapd.trace.BAK2.

You can add scripts or programs to those displayed when you add a crontab entry by putting them in the /usr/OV/cron directory. Any executable file in the /usr/OV/cron directory appears in the selection list of actions.

Attention: Because the /usr/OV/cron directory is part of the Tivoli NetView program, if the Tivoli NetView program is removed from the system, all files in the /usr/OV/cron directory may be lost. To avoid losing your executable files, either save them through the Server Setup application before you remove the Tivoli NetView program or create a symbolic link to the /usr/OV/cron directory. For example, the following command causes the date command to appear in the list of actions:

```
ln -s /usr/bin/date /usr/OV/cron/date
```

Creating a crontab Entry

To set a crontab entry using the Server Setup application, follow these steps:

1. Access the Server Setup application by entering **serversetup** on the command line.

2. Select **Maintain –> Manage crontab entries –> Add crontab entry**.
   The Add crontab entry dialog as show in Figure 12 is displayed.

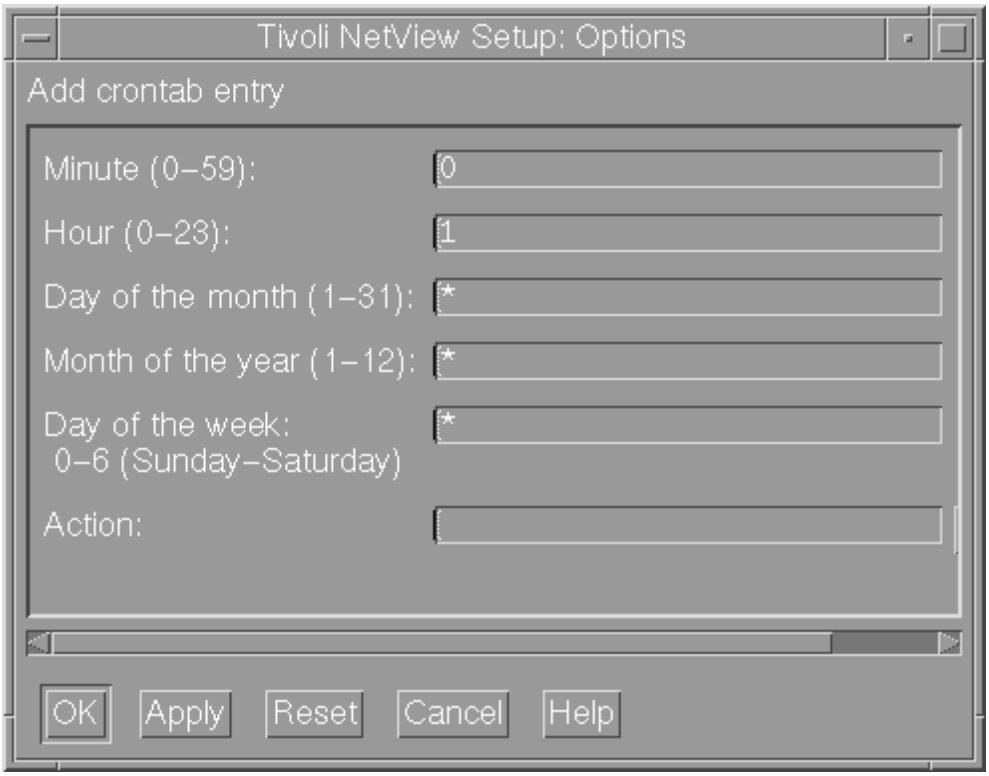

Figure 12. Add crontab Entry Dialog

3. Change the defaults in the entry fields.
4. Enter the name of the appropriate script in the Action field, or click the button beside the field and choose a script from the list.
5. Click **OK**.
   The information is processed, and a crontab entry is added.
6. Click **Close**.
   The dialog closes.

To see or clear a previously set crontab, select **List crontab entries** or **Remove crontab entry** from the **Manage crontab entries** menu.

Example of a Crontab Entry

The following is an example of a crontab entry that was set for the netmon.trace_Maint script:

```
59  1  *  *  1,3,5  /usr/OV/cron/netmon.trace_Maint
```

The netmon.trace_Maint script is executed at 1:59 a.m. every Monday, Wednesday, and Friday.

Maintaining Data Collection Files

Maintain the following data collection directories to ensure they do not use up all available disk space:

- /usr/OV/databases/openview
- /usr/OV/databases/tralertd
- /usr/OV/databases/snmpCollect
- /usr/OV/log

These directories continue to grow as long as you are collecting data. To prevent this problem, do one of the following:

- Create a crontab entry, using the Server Setup application to periodically remove log, trace, and collector files.

  See “Running Commands at Preset Times” on page 63 for information about how to create a crontab entry.

- Configure the trapd daemon to automatically archive trapd log data.

  See “Maintaining the trapd.log File” on page 61 for information about how to configure the trapd daemon.

- Maintain the databases.

  See “Maintaining the Databases” for information about how to maintain the databases.

- For the snmpCollect directory only:
  - Reduce the polling intervals.

    Select **Tools –> Data Collection & Thresholds: SNMP** to decrease the polling interval. If you only want to check thresholds, do not store the data.
  - Remove the last 100 entries of a file in the snmpCollect directory using the following commands:

    ```
    snmpColDump -tTI /usr/OV/databases/snmpCollect/file | \
       awk -F\t '{printf("%d\t%d\t%s\t%1g\n", $4, $5, $6, $3)}' | \
       tail -100 > /tmp/save
    snmpColDump -r /tmp/save /usr/OV/databases/snmpCollect/file
    ```

    Where *file* is the name of the collection file in the snmpCollect directory.

  C5 databases hold data used to map event frequency from APM submap icons. The C5 directory contains one file for each event type per node.

- For the C5 directory only you can:
  - Delete all these files safely if you do not need to map the traps.
  - Decrease the number of lines kept via the daemon option.
  - Delete the last **n** entries of each file.
  - Delete files that have not been updated recently.

Maintaining the Databases

Databases continue to grow and consume file system space as long as you are collecting data. This can adversely affect performance. To regain file system space, use one or more of the following methods:

- Resolve inconsistencies between the IP topology database and the database maintained by the **ovwdb** command for the graphical user interface. Resolving inconsistencies can result in deleting unneeded objects from the IP topology database.

  See “Resolving Database Inconsistencies” on page 66 for more information.

- Compress the IP topology database.

  Compressing the IP topology database can be effective in regaining file system space if a significant number of objects have been deleted from the IP topology database, either through normal editing or by using the **ovtopofix** command.

  See “Compressing the IP Topology Database” for more information.
- Clear the databases.

  Because customization data is lost when you clear the databases, you should clear the databases only if you were unable to regain file system space after trying the preceding methods.

  You can clear the following databases:
  – tralertd
  – snmpCollect
  – topology
    - ovwdb
    - topo
    - mapdb
    - defmap
    - gtmdb

  See “Clearing Databases” on page 67 for more information.

Resolving Database Inconsistencies

You can use the Server Setup application to resolve inconsistencies between the IP topology database and the database maintained by the ovwdb daemon for the graphical user interface and to resolve inconsistencies between the map database (mapdb) and the ovwdb database. The Server Setup application uses the ovtopofix and ovmapcount commands, respectively, to resolve these database inconsistencies.

To resolve database inconsistencies, with root permissions, follow these steps:

1. Exit all graphical user interfaces, including those on the client machines.
2. Enter **serversetup** on the command line to access the Server Setup application.
3. Select **Maintain –> Resolve database inconsistencies**.

   The Resolve database inconsistencies dialog is displayed.
4. Changes the entry fields.

   Refer to the online help for information about the entry fields.
5. Click **OK**.

   A warning dialog is displayed.
6. Click **OK**.
7. Click **Close**.

   The dialog closes.

Compressing the IP Topology Database

To compress the IP topology database using the Server Setup application, follow these steps:

1. Enter **serversetup** on the command line to access the Server Setup application.
2. Select **Maintain –> Compress the IP topology database**.

   The Compress the IP topology database dialog is displayed.
3. Select **yes** in the OK to continue? field.
4. Click **OK**.

   A warning dialog is displayed.

5. Click **OK**.

   The IP topology database is compressed by reading in all data from the IP topology database, truncating the private IP topology database, and rewriting the data.
6. Click **Close**.

   The dialog closes.

Clearing Databases

Attention: When you clear the databases, customization data is lost.

To clear the contents of the databases, follow these steps:

1. Enter **serversetup** on the command line to access the Server Setup application.
2. Select **Maintain –> Clear databases**. Then, select one of the following:
   - Select **Clear object/topology/map databases, save customizations** to remove the data from the /usr/OV/databases/openview/topo directory, except the APM definitions, SmartSet definitions, and master polling and discovery settings.
   - Select **Clear object/topology/map databases, remove customizations** to remove all the data from the /usr/OV/databases/openview/topo directory.
   - Select **Clear tralertd database** to remove all the files from the /usr/OV/databases/tralertd directory.
   - Select **Clear snmpCollect database** to remove all the files from the /usr/OV/databases/snmpCollect.

   A warning is displayed.
3. Click **OK**.

   The selected database is cleared.
4. Click **Close**.

   The dialog closes.

Note: When you use the Server Setup application to clear openview databases, all the daemons are stopped but not restarted. When you use the Server Setup application to restart map generation, all the daemons are stopped and restarted.

Deleting Unused Entries in the ovsuf File

If you have root permissions, you can delete entries from the ovsuf file. The ovsuf file contains the configuration information that prompts the startup process to start the specified daemons. When you set options for the Tivoli NetView daemons, entries are added to the ovsuf file or marked as unused. The entries that are marked as unused begin with the number `1:` in the ovsuf file. The `1:` prevents the startup process from starting this particular entry. You can delete these entries to prevent the file from becoming too large. Use the Server Setup application to delete unused entries (entries beginning with the `1:`).

See “Deleting Entries in the ovsuf File Using the Server Setup Application” on page 68 for the steps to accomplish this task.

Example of ovsuf File

Following is an example of the ovsuf file:

```
1:ovwdb:/usr/OV/bin/ovwdb:OVs_YES_START::-O:OVs_WELL_BEHAVED:15:
0:trapd:/usr/OV/bin/trapd:OVs_YES_START:::OVs_WELL_BEHAVED::
0:pmd:/usr/OV/bin/pmd:OVs_YES_START::-Au:OVs_WELL_BEHAVED::
```

Deleting Entries in the ovsuf File Using the Server Setup Application

To delete unused entries in the ovsuf file, follow these steps:

1. Enter **serversetup** on the command line to access the Server Setup application.
2. Select **Maintain –> Reset startup files –> Remove unused records from ovsuf startup file**.

   A warning is displayed.
3. Click **OK**.

   Unused records are deleted.
4. Click **Close**.

   The dialog closes.

Removing Old Snapshots

To free memory and improve system performance, select **File –> Map Snapshot –> Delete** from the Tivoli NetView graphical user interface menu bar. This option enables you to remove snapshots you no longer need.

Alternatively, you can remove snapshots using commands or using the Server Setup application.

Removing Snapshots Using the Command Line

To use commands to remove snapshots, follow these steps:

1. List all current maps by entering:

   /usr/OV/bin/ovmapdump -l

   This produces a list similar to the following example:

   ```
   MAP            PERMS  CREATION TIME              COMMENTS
   default        R/W    Mon Apr 27 11:52:32 1992
   Example Map 1  R/O    Wed Apr 22 12:05:48 1991   example map
   Example Map 2  None   Tue Apr 28 14:37:49 1992
   ```
2. List all available snapshots for a given map by entering:

   /usr/OV/bin/ovmapsnap -l -m *mapname*

   Where *mapname* is the name of the map.

   This produces a list similar to the following example:

   ```
   NAME      CREATION TIME               COMMENTS
   Testing   Fri Apr 24 12:05:48 1992    testing ovmapsnap
   ```
3. Enter the following to delete a snapshot:

   /usr/OV/bin/ovmapsnap -d -n "*snapshot***" -m** *mapname*

If you want to automatically delete the oldest entry, use the following command to set up a crontab entry:

/usr/OV/bin/ovmapsnap -d -f -m mapname

See the **ovmapsnap** man page for more information.

Removing Snapshots Using the Server Setup Application

To remove old snapshots using the Server Setup application, follow these steps:

1. Enter **serversetup** on the command line to access the Server Setup application.
2. Select **Maintain –> Manage map snapshots –> Remove map snapshot**.

   The Remove map snapshot dialog is displayed.
3. In the Map Name field, type the name of the map or click the button beside the field and select from the list.
4. In the Map Snapshot Name field, type the name of the map snapshot or click the button beside the field to select the map snapshot name.
5. Click **OK**.

   The information is processed, and the map snapshot is deleted.
6. Click **Close**.

   The dialog closes.

Cleaning Up the ORS Database

When the orsd daemon removes entries from its ORS database, it does not physically delete them, but marks the entries as having been deleted. The reason for this is to avoid rewriting the database each time an entry is removed, making the removal process faster.

However, keeping deleted entries in the database over a long period of time can waste file space and reduce the performance rate of database inquiries. Therefore, the orsd daemon is capable of removing the deleted entries from the database. This is known as **garbage collection**.

By default, the orsd daemon will periodically remove the deleted entries. Or, if you have root permissons, you can initiate garbage collection by using the **ovorsutil -g** command.

You can use the Server Setup application to set options on the orsd daemon to automatically do periodic garbage collections. You can set the following values:

- How often the orsd daemon checks to see if there are deleted entries in the database.
- The percentage of garbage that must be in the database before a garbage collection is performed.

See “Event and Trap Processing Daemons” on page 32 for information on setting these orsd daemon options.

Appendix A. Memory, Paging Space, Tuning, and Sizing Considerations

This section contains information that will help you estimate the minimum amount of memory that you will need for your Tivoli NetView server and Tivoli NetView client systems. Because insufficient memory in your Tivoli NetView system will have a significant negative impact on system performance, you should ensure that you have adequate memory.

This section also describes how to determine the correct size and placement of system paging space. It also offers tuning suggestions for Tivoli NetView. Finally, it suggests ways to classify managed networks based on size and workload to aid in planning.

Disclaimer: The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer's ability to evaluate and integrate them into the customer's operational environment. While IBM and Tivoli may have reviewed each item for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere.

Customers attempting to adapt these techniques to their own environments do so at their own risk. Any performance data contained in this document was determined in a controlled environment; therefore, the results that may be obtained in other operating environments may vary significantly. Users of this document should verify the applicable data for their specific environment.

You should refer to the latest release notes for any updates to the information about memory, tuning, and other such requirements. Be careful to allocate adequate memory for your operating system configuration to accommodate Tivoli NetView as well as any other applications that are running on the system.

Performance evaluations on all UNIX platforms yield similar results. Memory and paging space requirements have been derived based on detailed AIX testing and analysis. When using Solaris, some allowance should be made for small variations in memory and paging space needs.

Estimating Memory Requirements

To estimate the memory requirements for Tivoli NetView, you should answer the following questions:

- What is the size of the network that the Tivoli NetView server will discover? As the size of the discovered network grows, so does the amount of memory that Tivoli NetView requires. Tivoli NetView memory estimates are done on the basis of *objects*, which are the internal representations of real-world items. How to count objects is discussed in the sections that follow.
- How many concurrent Tivoli NetView operator graphical user interface sessions will there be? Will these graphical user interface sessions be running on the server, client or as a Web client? As the number of graphical user interface operators increases, the memory needed to support them increases.
- What other applications do you plan to place on your Tivoli NetView server or client system?

After you have determined the answer to each of these questions, use the memory tables to identify how much memory you need for each system.

Determining the Size of the Network

When the Tivoli NetView program discovers a network, it represents the network devices as objects. Objects are the internal representation of real-world items. The key network objects are:

- IP Networks
- IP segments
- IP Nodes (for example IP-addressable workstations, routers, switches, servers, IP-addressable hubs, and so on)
- IP Addressable interfaces (network adapters found in workstations and adapters found in a router, for example)

To determine the number of objects contained in the topology database, use **one** of the following methods:

1. Count the number of interfaces and multiply this number by 2.5.
2. If you have not installed the Tivoli NetView program, add the number of nodes, IP-addressable interfaces, networks, routers (be sure to count each interface), and segments. Obtain this number by actual counts or estimates. For a simple estimate, count the nodes, routers and interfaces.
3. If you have Tivoli NetView installed, determine the number of actual objects by using the **/usr/OV/bin/ovtopodump -l** command. The object count is the sum of Networks + Segments + Nodes + Interfaces. Note that the count for Gateways is not included.

Determining the Number of Operators

For an estimate of the number of operators required, determine the number of graphical user interface sessions that will be placed on each Tivoli NetView server and each Tivoli NetView client system.

Determining Memory Requirements for Additional Applications

To determine memory requirements for additional applications that you are running with Tivoli NetView, consider not only the memory that each application requires, but also consider the additional memory Tivoli NetView requires to support the application.

The more information you have about an application and its use of Tivoli NetView services, the more accurate this estimate will be. Often, the documentation for an application does not specify its requirements on Tivoli NetView services. If this is the case, add the general memory estimates given in the product documentation to your estimate for Tivoli NetView memory. If you have knowledge of the application's use of Tivoli NetView services, the following text gives some insight to how to improve on the memory estimate.

To estimate the amount of memory the application requires, determine the memory that is required for the application's processes based on those elements that are significant for the application. Typically, an application has at least one daemon process and a graphical user interface process for each operator. You should refer to the appropriate vendor documentation for memory requirements for the application you are using.

Next, add the additional memory that Tivoli NetView processes require to support the application. Vendor applications can use Tivoli NetView APIs to increase the amount of information that is stored in the Tivoli NetView databases. Increasing the information in the databases increases the Tivoli NetView memory requirements.

For example, suppose that you have an application that relies on a user ID discovery process to discover workstation user IDs with all their associated data fields (such as, user name, user ID, host node name, office phone number, e-mail address, manager name, manager's e-mail address, and accounting number). Suppose also that this application uses a topology process to store the information in the Tivoli NetView **ovwdb** database, and it includes a graphical user interface process for each operator, which displays a map of the user IDs and their relationships (for example, user IDs on a host node). This application increases Tivoli NetView's memory requirements because there is an increase in objects in the **ovwdb** object database. The user ID discovery process, the topology process, and the map process for each operator graphical user interface session also add to the total memory requirement.

Computing Memory Needs Based on Object Count

This section provides a simple table for estimating memory requirements on the Tivoli NetView server or client. Using the object count you have established using the methods above, and the number of graphical user interface sessions on either the server or client system, scan the table to determine the memory needs.

Note: The memory sizing estimates in this table do not account for Web clients. Refer to the release notes for the latest information.

Table 8. Tivoli NetView Server and Client Memory Sizing Estimates in Megabytes

Objects	Server + 1	Client + 1	NextOpr
5000	128	64	21
10000	129	64	31
13000	153	64	38
15000	168	64	42
20000	207	69	53
25000	245	79	63
30000	284	90	74
35000	323	101	85
40000	361	111	95
45000	400	122	106
50000	438	132	116
55000	477	143	127
60000	516	154	138
65000	554	164	148
70000	593	175	159
75000	632	186	170
80000	670	196	180
85000	709	207	191
90000	747	217	201

Table 8. Tivoli NetView Server and Client Memory Sizing Estimates in Megabytes (continued)

Objects	Server + 1	Client + 1	NextOpr
95000	786	228	212
100000	825	239	223

Example 1

A Tivoli NetView server is managing a network of 35,000 objects, with one local operator and two additional operators using Telnet sessions to connect to the server. For this network, the memory is estimated as 323 + 2(85) = 493 MB. In this case, the approach is to round up to the next size of memory module (in this case 512 MB).

Note that this is a minimum estimate. It does not include other application memory requirements. Furthermore, the user activity model is limited to simple viewing of the topology display and events log.

Example 2

A second example is provided in Figure 13, illustrating primary and backup Tivoli NetView servers with a total of five operators supported and one Tivoli NetView client system supporting three operators. This example is for 50,000 objects.

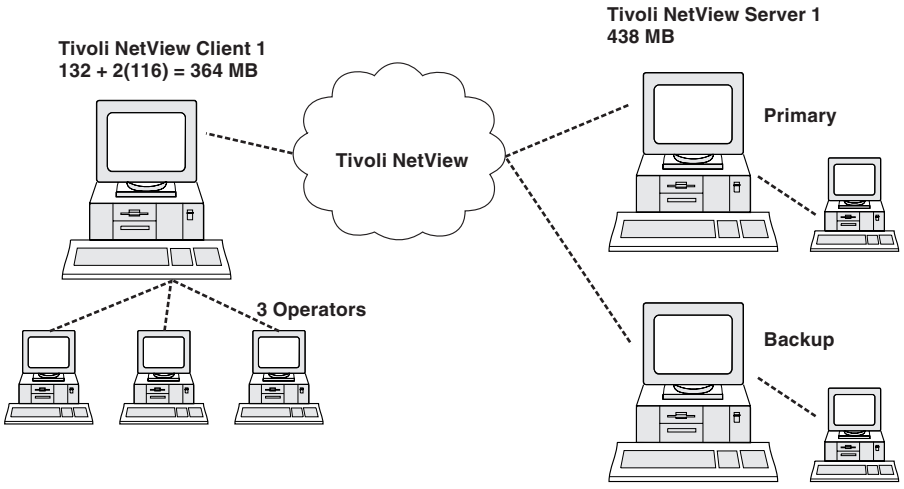

Figure 13. Memory Sizing Formula Example (50,000 Object Network)

Additional Memory Considerations

Here are additional considerations to keep in mind when you estimate memory requirements:

- When using the memory table in the preceding section, remember that its calculations represent a **minimum** estimate. It would be wise to add memory to the estimate.
- Do not overlook the memory requirements of other applications on your Tivoli NetView server.
- The estimates for operator memory are based on a minimal operator activity. Other activities like graphical display of historical data (using xnmgraphs, which require about 1 MB each) will add to the memory requirement.
- Plan for growth of your network that can be accommodated by the initial memory placed in the system.

- Select memory in the largest DIMM units available for your system to allow for memory expansion without wasting smaller sized DIMMs in the upgrade.
- Monitor memory and system resource utilization with a routine program of performance data collection.
- The memory installed in X-stations is unrelated to the memory requirements for the graphical user interface processes running on the Tivoli NetView server or client, as described above. The X-station memory is used to store the X graphics for the X-station display and processor.
- Experience from past releases of Tivoli NetView indicate that you need a minimum of 24 MB of memory available in the X-station. Although you can use less memory, 24 MB reduces the probability of memory contention problems. On most X-stations, when memory contention occurs, X closes an application, which usually causes the application to close. X-stations have not been used for some time in these investigations.
- Using an X-emulator on a Windows/NT system as a Telnet operator has the same memory costs on the Tivoli NetView server or client as using a UNIX workstation for the same task.
- Background graphics in submaps can require a considerable amount of extra memory; 5 to 10 MB is not uncommon.

Miscellaneous Considerations

Here are some miscellaneous considerations:

- Tivoli NetView can take advantage of multiple processors on a symmetric multiprocessor (SMP) machine.

  A SMP machine is a particularly good choice for a system that is in use as a Tivoli NetView client and supporting multiple graphical user interface sessions. A SMP system is also an excellent choice for the Tivoli NetView server because such a system can more effectively service key components such as status monitoring, trap processing, database service, and local graphical user interface support processes.
- Consider a second, identical backup system for your Tivoli NetView server that can provide quick take over if your primary Tivoli NetView server experiences problems.
- An attractive choice for a system that could be used as a combination Tivoli NetView client and operator system would be one of the entry-level, dual-processor systems discussed in the release notes. Using an entry-level system eliminates the expense of a separate operator machine that is used to connect to a larger system supporting multiple graphical user interface sessions, removes the need for a large client system, and provides for maximal workload distribution.
- When defining the connection between the Tivoli NetView server and client, you can use local or NFS-mounted connections. The NFS option provides the easiest way to manage the consistency between the map view on the Tivoli NetView server and the client. Note that for NFS supported client systems, the logon and synchronization times for graphical user interface sessions will be longer when compared to Telnet sessions to the Tivoli NetView server
- Longer synchronization times are required for transferring topology information (often a substantial amount of data) from the Tivoli NetView server to the client. For this reason, you should not place the client system on a slow network connection to the Tivoli NetView server.
- Select the largest L2 cache option available for the CPU.

Paging Space Guidelines

Paging space requirements are unique for each system, dependent upon such variables as which applications are running and the number of active users.

Remember the following guidelines when you evaluate how much paging space you need.

- Some systems with large amounts of memory do not need large amounts of paging space. If a machine is in a persistent (data files) storage environment (that is, the machine has a few small programs and a large amount of data in files that reside in the memory-based disk cache), you might not require paging space equal to the amount of memory. The files in the memory-based disk cache will not be paged out to paging space.
- Too much paging space is not necessarily good because unused paging space is not available for other uses. However, not having enough paging space can cause problems on your system. Strive to achieve balance between these two opposing goals.
- For systems with 256 MB of memory or less, a good starting point to determine the amount of paging space is to multiply the amount of memory by 2. To determine the amount of paging space required for systems with more than 256 MB of memory, use the following formula:

  ```
  Paging Space (in MB) = 512 + (memory - 256) * 1.25
  ```

  For example:

If memory (in MB) is...	Paging space (in MB) should be at least...
64	128–192
128	256
256	512
512	832
1024	1472

Creating or Enlarging Paging Space

Follow these guidelines when creating or enlarging the amount of paging space:

- Do not place more than one paging space logical volume on a physical volume.

  All processes that are started during the boot process are allocated paging space on the default paging space logical volume (hd6).

  After the additional paging space logical volumes are activated, paging space is allocated in a series in 4 KB blocks, selecting the next active page space using a serial sequential algorithm.

  If you have paging space on multiple physical volumes and you put more than one paging space on a physical volume, you will no longer be spreading your paging activity evenly over the multiple physical volumes. That is, the physical volumes with more than one paging space will have more total paging activity.
- Do not extend a paging space logical volume onto multiple physical volumes.

  If a paging space logical volume is spread over multiple physical volumes, you will not be spreading your paging activity across all of your physical volumes. If you want to allocate space for paging on a physical volume that does not already have a paging space logical volume, create a new paging space logical volume on that physical volume.

- Make each paging space logical volume approximately equal in size.

  If you have paging spaces of different sizes, when the smaller ones become full, you will no longer be spreading your paging activity across all of your physical volumes.
- Avoid putting a paging space logical volume on the same physical volume as a heavily active logical volume, such as that used by a database.
- It is not necessary to put a paging space logical volume on each physical volume.
- For best system performance, place paging space logical volumes on physical volumes that are each attached to a different disk controller. Spread the paging activity across as many disk controllers as possible.
- If you have several page spaces, each on a different disk drive, you must ensure that you deactivate any active paging space on a disk drive before removing the disk drive. If a disk drive containing an active paging space logical volume is removed from the system, the system will fail.

Indicators That More Paging Space is Required

You might need more paging space for any of the following reasons:

- If processes die due to having received signal 33 (SIGDANGER)

  To verify this, use the **errpt** command to view the system error log entries. When the system sends a signal 33, it indicates that the system has only about 2 MB of free page space. Shortly after the system sends a signal 33, the system starts killing the most current processes.
- If any of the following messages are displayed:

```
INIT: Paging space is low

ksh: cannot fork no swap space

Not enough memory

Fork function failed

fork () system call failed

Unable to fork, too many processes

Fork failure - not enough memory available

Fork function not allowed. Not enough memory available.

Cannot fork: Not enough space

signal 33 received

SIGDANGER received
```

- If the average of the percent used, or the percent used, is greater than 80 percent

The **iostat**, **vmstat**, and **lsps** commands can help you determine if you want to make changes regarding paging space logical volumes.

Tuning Tivoli NetView

This section contains miscellaneous tuning suggestions. In a normal environment, Tivoli NetView is configured for a good balance of performance and information. For very large networks, or in an environment where it is critical to achieve optimal performance, follow the guidelines in this section.

- Filtering traps from display on the Event Display is effective in reducing CPU consumption. While filters do add cost to the processing of traps (the trap must be checked to see if the filter should apply), the cost of checking the filter is far less than the cost of displaying the trap on the Event Display. Carefully review the traps presented at the Event Display and develop filters to reduce the display of traps that you determine are not of interest to your environment.
- There is no performance difference between the use of the card and row display option on the Event Display.
- The cost of trap transmissions to the Tivoli Event Console is minimal. Define rules for transmission of traps from Tivoli NetView to the Tivoli Enterprise Console as the trap processing and transmission capacity of Tivoli NetView is higher than the Tivoli Enterprise Console reception and processing capacity.
- Ensure that trap definitions provided by other applications placed on your Tivoli NetView server are properly defined to Tivoli NetView. The cost of determining that a trap is undefined is substantially higher than the identification of a known trap.
- Set the cache size for the Tivoli NetView ovwdb daemon to be greater than the object count. Use the **/usr/OV/bin/ovobjprint -S** command to determine the number of objects in the database. Increase the ovwdb cache size using the Server Setup application (**Configure –> Set options for daemons –> Set options for topology, discovery, and database daemons –> Set options for ovwdb daemon**).
- For environments with high trap rates (more than 5 per second), increase the Tivoli NetView trapd queue size to 15,000. Values from 500 to 50,000 have been confirmed to be stable. Large queues prevent trap loss, but can buffer a lot of traps during a network storm that must be processed. To increase the trapd queue size, use the Server Setup application (**Configure –> Set options for daemons –> Set options for event and trap processing daemons –> Set options for trapd daemon**).
- Displaying large numbers of events can be CPU intensive. For operators who do not require the Event Display all the time, Tivoli suggests not starting it up by default. To disable initial startup of the Event Display, edit the /usr/OV/registration/$LANG/ovsnmp/nvevents file and remove the -Initial flag from the following line:

  ```
  Command -Shared -Initial "$[nvevents:-/usr/OV/bin/nvevents]"
  ```

- SmartSets provide a powerful tool for gathering resources with some common characteristic into a group. They can provide a way to segment different populations of resources for such values as different monitor interval and time out values. SmartSets extend the initialization period for the user interface (the synchronization time) and Tivoli NetView startup. However, this is usually not a factor as the number of SmartSets is small or the number of items in a SmartSet is small. Each Mid-Level Manager will result in a SmartSet (if you are using APM).
- Consider these DNS guidelines.
  - Tivoli NetView performs many DNS lookup calls.
  - Keep the size of your **/etc/host** file small.

  - Consider the placement of the DNS relative to Tivoli NetView. Place the primary DNS near Tivoli NetView in the network.
  - Consider using a secondary DNS on the Tivoli NetView server. Experience indicates that a secondary DNS on the Tivoli NetView server has a small CPU cost.
- Use the following guidelines for efficient seed file discovery:
  - Periodically run **ovtopofix** to clean out obselete references.
  - Use IP addresses in your seed file.
  - Consider a stopper address as the last entry to stop discovery.
  - Consider customization of **/usr/OV/conf/oid_to_type**. For example, designate **DEFAULT_IP: : :U** as last line of **/usr/OV/conf/oid_to_type** to set nodes not running SNMP as unmanaged and Tivoli NetView will not deal with them again.
  - Work to perfect the seed file. Discover the network, adjust the seed file, delete the database, and then try the discovery again. Seed files are an excellent technique to bound discovery (saves memory).
  - Note that large seed files can add to the Tivoli NetView startup time.
- To increase the status monitoring response time for large networks, consider increasing the netmon ping queue size to a maximum of 32 (the default is 16). To increase the ping queue size:
  1. Add the **-q32** flag to the **/usr/OV/lrf/netmon.lrf** file.
  2. Type the following commands:

     ```
     /usr/OV/bin/ovstop netmon
     /usr/OV/bin/ovaddobj /usr/OV/lrf/netmon.lrf
     /usr/OV/bin/ovstart netmon
     ```

  To accelerate the initial SNMP Configuration Update and Discovery for large networks, you might also consider increasing the netmon SNMP queue size to a maximum of 32 (the default is 16).

  Be aware that increasing the SNMP queue size will significantly increase the SNMP traffic on your network and system. This can degrade the performance of your management station and network. Monitor your system and network after increasing this queue to determine the optimal value for your environment.

  To increase the SNMP queue size:
  1. Add the **-Q***num* num flag (where *num* is the queue size) to the **/usr/OV/lrf/netmon.lrf** file.
  2. Type the following commands:

     ```
     /usr/OV/bin/ovstop netmon
     /usr/OV/bin/ovaddobj /usr/OV/lrf/netmon.lrf
     /usr/OV/bin/ovstart netmon
     ```

Network Sizing Guidelines

This section contains general definitions of networks based on size and workload. These definitions are based on laboratory performance studies and customer observations.

The information in this section should be considered a general guideline to help you select systems with appropriate capacities. The guidelines assume that the default Tivoli NetView database is in use, that you have used the tuning described above, and that Tivoli NetView is the only significant application on the system. They

should not be taken as a performance guarantee.

Table 9. Characteristics of Sample Networks Based on Size

Network Size	Characteristics
Small network	• 1000 to 20,000 objects • 1 or 2 Telnet operators • Status monitoring with 5 to15-minute polling • Few or no SNMP collections • No other applications on the system • 5 to 10 traps per minute This environment has excellent network response times (4–10 MS).
Medium network	• Around 30,000 objects • 2 or 3 Telnet operators • 1 or 2 Web operators • Status monitoring five-minute polling • Some SNMP collections • No other applications • 20 to 30 traps per minute • Approximately 300 routers This network has good network response times (10–70 MS).
Large network	• More than 40,000 objects • 3,000 to 4,000 routers • 3 to 10 Telnet operators • 1 to 5 Web operators • A large system used as the Tivoli NetView client to support most of the operators • Backup Tivoli NetView server present • Status monitoring at 5, 15, and 30–minute intervals with polling set up for three populations • SNMP collections at 15 minute intervals to 4000 nodes requesting one MIB value • 1 to 2 traps per second This network has good response times. Deployment of Mid-Level Managers could be considered for the future of this network.

General recommendations include:

- Fastest possible SMP machine (2-4 CPUs) for your Tivoli NetView server or Tivoli NetView client machines
- PCI network adapter card
- Multiple hard disks
- Adequate memory
- Design for introduction of distributed components (such as Mid-Level Manager, client/server, operator types, and placement)
- Backup the Tivoli NetView server

Some additional sizing information is available in the operating-system specific appendices of this book. Contact your Tivoli support representative for suggestions about a suitable machine configuration for your environment.

Appendix B. Additional Notes for AIX

This appendix contains additional information for users of Tivoli NetView on an AIX platform.

Mounting a CD-ROM on AIX

Following is additional information about mounting a CD-ROM on AIX:

1. Create a CD-ROM file system. If this has already been done, proceed to step 2. To create a CD-ROM file system:
   a. Start SMIT using the following command:
      ```
      smit crfs
      ```
   b. Select **Add a CDROM File System** from the SMIT menu. Complete the fields in this dialog.
      The DEVICE Name is likely to be cd0 (check for the file **/dev/cd0**; if it does not exist, determine the name of your cdrom device).
      A suggested name for MOUNT POINT is **/cdrom**. Ensure that the directory specified exists and that it is an empty directory.
   c. Click **OK** to add the CD-ROM file system and then exit from smit.
2. Insert the CD-ROM in the drive and execute the following command:
   ```
   mount  /cdrom
   ```
   Where */cdrom* is the CD-ROM file system.

   Note: To unmount a CD-ROM on AIX, execute the command:
   ```
   unmount  /cdrom
   ```

High Availability Cluster Multi-Processing Servers on AIX

Tivoli NetView can be used with High Availability Cluster Multi-Processing (HACMP) servers with the following recommendations:

- Put /usr/OV on the fail-safe file system.
- Have both IP and MAC address takeover.
- Install Tivoli NetView on both machines with the hostname set to the one corresponding to the shared IP address.

Even though /usr/OV will be overwritten, this is necessary for the surrounding links and modifications to the nonshared system files/directories to occur. Because /usr/OV is shared, a backup of /usr/OV should be made prior to deinstalling, so that it can be restored for the deinstallation of the other machine to occur without a problem.

Use the following startup/shutdown scripts:

```
start_netview script:

# Mount shared directory
mount /usr/OV
# Source in Tivoli info for appropriate Library paths
. /etc/Tivoli/setup_env.sh
# Set hostname to shared IP addresses hostname
hostname hacmp5
# Export subdirectories needed by client
mknfsexp -d '/usr/OV/conf' -t 'ro' -r \
```

```
'hacmp77.net.com' -N
mknfsexp -d '/usr/OV/databases/snmpCollect' -t 'ro'\
-r 'hacmp77.net.com' -N mknfsexp -d
'/usr/OV/databases/openview/mapdb' -t 'rw' -r \
'hacmp77.net.com' -N mknfsexp -d
'/usr/OV/databases/openview/defmap' -t 'rw' -r \
'hacmp77.net.com' -N
# Set display and start netview
# Note: If it is to be used as a GUI-less server then
# the DISPLAY doesn't need to be set and /etc/netnmrc
# should be run.
export DISPLAY=:0.0
/usr/OV/bin/netview
# End of script
```

```
stop_netview script:

# Stop daemons/windows so that the /usr/OV can be unmounted
/usr/OV/bin/nv6000_smit stopall forceall
/usr/OV/bin/nv6000_smit APPLCLEANUP ovw_binary nvauth nvsec_admin
/usr/OV/bin/nettl -stop >/dev/null 2>&1
/usr/OV/bin/ovstop nvsecd >/dev/null 2>&1
# Remove /usr/OV subdirectories from NFS exports list
rmnfsexp -d '/usr/OV/conf' -N
rmnfsexp -d '/usr/OV/databases/snmpCollect' -N
rmnfsexp -d '/usr/OV/databases/openview/mapdb' -N
rmnfsexp -d '/usr/OV/databases/openview/defmap' -N
# Unmount /usr/OV
cd /
umount /usr/OV
# End of script
```

If client/server is to be used, it is important to set the Major Number for the shared volume group consistently. For information about about the network file system (NFS) and HACMP, refer to the *IBM AIX High Availability Cluster Multi-Processing/6000 Administrator's Guide*.

Also, when configuring the client, you should use the hostname of the shared IP address for the server.

Tuning Suggestions for AIX Systems

This section describes miscellaneous AIX tuning recommendations to improve your performance.

- Be aware of these changes to the AIX buffers for network adapter cards:
  - AIX 4.3.1 provides larger buffers over past AIX levels.
  - PCI adapters allow larger buffers than MCA adapters.

  Tivoli recommends that you set transmit and receive buffers for your network adapter card as large as allowed for your Tivoli NetView server and Tivoli NetView client systems. Follow these steps to increase the size of these buffers. This example assumes that the adapter is active and for the **tr0** token ring.

  1. Enter **ifconfig tr0 detach**.
  2. Make the adjustments using smit. The smit fast path commands are **smit tradap** or **smit ethernet**.
  3. Enter **ifconfig tr0 hostname up**.
- The TurboDatabase speed option has been shown to reduce I/Os and improve network discovery response times. Refer to "Appendix D. NDBM Database Enhancements in Tivoli NetView Version 5.1 (AIX only)" on page 89 for more information on this utility.

- Increase the **Processes per user** to 1024. Use the smit utility to make this change.
- The ARP Cache is used to translate IP addresses to hardware addresses on the same subnet. A good example is the default router address. The default (175 for AIX Version 4) is usually acceptable.
  - Use **arp -an | wc -l** to check cache usage counts.
  - Use **no -a** to view the **arptab_nb** (number of "buckets") and **arptab_bsiz** (size of "buckets") settings.

    The cache size is equal to the product for these two values. Change with the **no -o arptab_nb=293** (prime #) command, add this change to the **/etc/rc.net** file (careful with placement), and confirm that the size is adequate.
  - The ARP cache is less interesting in routed networks.
  - The ARP cache is more interesting in switched networks.

Recommended AIX Machine Types

Following are some recommended machine types, based on the size of your network.

Table 10. Sample Machine Types

System Size	Machine Type
For a small network	• RISC/6000 43P-240 dual 233 MHz • RISC/6000 F50 dual 332 MHz
For a medium network	• RISC/6000 F50 4-way 332 MHz
For a large network	• Multiple systems • Server or client: RISC/6000 F50 or H50 4-way 332 MHz • Combination client and single operator: RISC/6000 43P- 260 dual 200 MHz • RISC/6000 H70 4-way 340 MHz

Refer to Table 9 on page 80 for definitions of small, medium, and large networks.

Individual machine models are frequently changed and updated by hardware manufacturers, the suggested machines were current at the time this material was prepared and should be considered as a class of systems with documented (using industry standard benchmarks) performance capabilities.

Tuning AIX for Tivoli NetView

For information about AIX tuning and Tivoli NetView, refer to "Appendix A. Memory, Paging Space, Tuning, and Sizing Considerations" on page 71.

Appendix C. Saving Files and Installation Entries

This appendix provides information about installation entries and saving files.

Saving Files

This section provides information about saving data files.

Saving Files Using the Tivoli Desktop (Version 5 and 6)

To save your Version 5.0, Version 5.1, or Version 6.0 data files using the Tivoli desktop, follow these steps:

1. Enter **tivoli** at the command line to access the Tivoli desktop.
2. For Version 5.0 or Version 5.1, double-click the policy region that contains the appropriate Tivoli NetView server. The Policy Region window is displayed.
3. Click and hold down the right mouse button on the server icon to display the menu.
4. Click **Maintain –> Backup Selective Data**.

   The Backup Selective Data dialog box is displayed.
5. Complete the following fields in the dialog box:
   - Directories to save:
   - Volume group for backup filesystem:
   - Replace existing backup data?

   Refer to the online help for information about these fields. See “Appendix E. Files That Migrate” on page 95 for descriptions of the categories of files that you can save.

Note: If you have installed the Tivoli NetView Framework 5.x to 6.0 patch, you are not able to back up your current Tivoli NetView Version 5.x installation using the Tivoli Desktop. Use the Tivoli NetView migration script to back up your data. See “Saving Files Using the Migration Script” on page 86 for information on how to use this script.

Saving Files Using Tivoli NetView Server Setup (Version 6 or Higher)

To save your data files using the Tivoli NetView Server Setup application, follow these steps:

1. Enter **serversetup** from the command line.
2. Click **Maintain –> Backup Selective Data**. The **Backup Selective Data** dialog box is displayed.
3. Complete the following information in the fields:
   - Directories to save
   - Volume group for backup filesystem:
   - Replace existing backup data?

   Refer to the online help for information about these fields. See “Appendix E. Files That Migrate” on page 95 for descriptions of the categories of files that you can save.

Note: To save files on a client, enter the **clientsetup** command in Step 1 above, and then follow Steps 2-3.

Saving Files Using the Migration Script

To save data files using the migration script, follow these steps:

1. Exit the graphical user interface if it is running.
2. Stop all the daemons.
3. Run the migration script by entering the following command:

   ```
   /usr/OV/install/tools/nvp.vXrY save
   ```

   Where *nvp.vXrY* is the script that corresponds to the version that you currently have installed, for example, nvp.v6r0. You will be prompted for the categories of the files you want to backup. See “Appendix E. Files That Migrate” on page 95 for descriptions of the categories.

   To backup all data from the Version 6.0 installation without being prompted for which categories to backup, use the following command:

   ```
   /usr/OV/install/tools/nvp.v6r0 save -p '/usr/OV/ALL' -vg 'none'
   ```

Note: For a client, the migration script is called nvpc.vXrY.

Installation Entries

The installation procedure adds the following entries for Tivoli NetView processes to the following files. These entries should not be changed.

Table 11. Installation Entries

Entry	Process	File
/etc/netnmrc	Background daemons	/etc/rc.tcpip (AIX only)
actionsvr 1670/tcp	actionsvr	/etc/services
C5_server 1668/tcp	C5d	/etc/services
cmot_manager 163/tcp	pmd	/etc/services
cmot_manager 163/udp	pmd	/etc/services
cmot_agent 164/tcp	pmd	/etc/services
cmot_agent 164/udp	pmd	/etc/services
gtmd 2112/tcp	gtmd	/etc/services
mgragentd 1.3.6.1.4.1.2.6.4.6 nv6000	mgragentd	/etc/snmpd.peer (AIX only)
nvcold 1664/tcp	nvcold	/etc/services
nvcorrd 1666/tcp	nvcorrd	/etc/services
nvlockd 1669/tcp	nvlockd	/etc/services
nvpagerd 1671/tcp	nvpagerd	/etc/services
nvsecd 1663/tcp	nvsecd	/etc/services
nvsecltd 1667/tcp	nvsecltd	/etc/services
nvtrapd-trap 162/tcp	trapd	/etc/services
nvtrapd-trap 162/udp	trapd	/etc/services
nvtrapd-client 1661/tcp	trapd	/etc/services

Table 11. Installation Entries (continued)

Entry	Process	File
`otmd 1672/tcp`	otmd	/etc/services
`ovtopmd 8888/tcp`	ovtopmd	/etc/services
`ovwdb 9999/tcp`	ovwdb	/etc/services
`pmd 2113/tcp`	pmd	/etc/services
`smux 1.3.6.1.4.1.2.6.4.6 nv6000`	mgragentd	/etc/snmpd.conf (AIX only)
`snmp 161/udp`	TCP/IP Agent	/etc/services
`smux 1.3.6.1.4.1.2.6.4.1 nv6000`	trapgend	/etc/snmpd.conf (AIX only)
`snmp 161/udp`	TCP/IP agent (snmpd process)	/etc/services
`trapgend 1.3.6.1.4.1.2.6.4.1 nv6000`	trapgend	/etc/snmpd.peers (AIX only)
`xxmd 3113/tcp`	gtmd	/etc/services

Appendix D. NDBM Database Enhancements in Tivoli NetView Version 5.1 (AIX only)

Improvements were made in the Version 5.1 release of Tivoli NetView to address these problems on the AIX platform:

- Some large routers can cause Tivoli NetView to create fields that are too large to fit into the object database.
- Some of the database files are difficult to back up because they appear to be much larger than their actual data content. (for example, the sparse file system problem).
- High CPU time is sometimes required for ovwdb.
- ovtopmd and netmon speed is sometimes unacceptable.
- ipmap and xxmap synchronization time is long.

To address these problems, two enhancements were made. First, the code in the ovwdb API code for the OVwDbFieldNameToFieldId() call will now cache field IDs locally. This should improve client/server performance and is transparent to the calling program. And second, the NDBM component, which is the basis for the object, topology, and map databases, was enhanced in several ways. The remainder of this appendix will describe these changes.

NDBM Component Overview

Each of the three main Tivoli NetView databases is a collection of several NDBM databases. NBDM is an expandable keyed hash table with the data residing on the disk. Each NDBM database is comprised of two files:

- A directory file
- A page file

The directory file has an extension of **.dir** and contains information that NDBM uses to index into the page file.

The page file actually contains the key and the data. The file is divided into pages where the data (key and value) is stored. The page number is calculated from the key and the bits in the directory file. Each page is currently 8 KB long.

Two optional files are defined for each NDBM database.

- A config file
- An overflow file

The config file contains parameters for NDBM and determines what new functions will be applied to this database. The file is created by the dbmcompress utility and is not intended to be modified in an editor. If the config file does not exist, NDBM uses defaults that match the old NDBM processing, so that, even with the new code, existing databases continue to work as usual.

The overflow file contains data for large values. This will enable data larger than 8 KB to be stored in the database.

These new files provide the following capabilities:

- Large values can be written to the overflow file rather than the page file. This has two benefits. First, it allows values larger than 8 KB to be stored in the database. Second, it greatly reduces the sparse filesystem problem.
- New hashing algorithms exist to help group related data items into the same area of the disk. This can reduce disk writes for database updates.
- A new option to start permanently buffering database updates can reduce both disk read and writes. For some databases, buffering will provide a significant performance boost only when combined with the new hashing algorithms.
- If the environment variable **NV6K_NDBM_DEBUG** is turned on, then all database activity will be logged to a **.trc** file. This can be useful for debugging and occasionally for tuning.
- With the dbmcompress utility, some database problems, can be corrected. This utility cannot resolve inconsistencies between databases, but it can remove NULL values and inaccessible data from an individual NDBM database.

New NDBM Utilities

Three NDBM utilities provide the capabilities described in the previous section: dbmcompress, dbmlist, and nvTurboDatabase. These utilities reside in the **/usr/OV/service** directory.

The dbmcompress Utility

The dbmcompress utility compresses an individual NDBM database. Since it compresses one database at a low level, it can complete its compression much faster than older database compression utilities. Compressing the value_info achieves most of the benefits of the **ovwdbdmap -c** command in about one-fifth the time. The dbmcompress utility is also used to transform databases. The transformation creates a config file and a database based on command line options to dbmcompress.

The dbmcompress command has the following syntax:

```
dbmcompress [-o -s -a -h d|o|n|r -m -b -v ] databasename
```

where:

–o Means that new database will use an overflow file for large items.

–s *n* Will overflow objects larger than *n*.

–a *n* Will append items in the overflow until file size reaches *n*.

–h d|o|n|r
Determines the hashing algorithm for the new database, where **d** = default, **o**=group by oid, **n**=group by name, **r**=group by oid (method 2).

–m *n* Allows the hashing to optimize for *n* megabytes of data for hashing algorithms other than **d**.

–b *n* Will cause the database to always buffer *n* pages.

–v Indicates verbose flag.

The dbmlist Utility

The dbmlist utility collects important information such as the configuration of a database, the count of items in the database, and the total size of the database.

The syntax for the dbmlist command is:

```
dbmlist [-belcnsSh -i 'val' -j 'val'] databasename
```

where:

–b Prints block information.

–e Prints empty blocks.

–l Prints sizes of keys and values.

–c Counts the objects.

–n Does not print the object data.

–s Prints the total size of the the data and keys.

–S Suppresses database error messages.

–h Prints the key in hexadecimal.

–i 'val'
Prints only keys where the first word matches 'val'.

–j 'val'
Prints only keys where the second word matches 'val'.

The nvTurboDatabase Script

The nvTurboDatabase utility processes all appropriate database files for a customer and runs dbmcompress to transform and compress the databases. Because some NDBM database files must trade off between speed and database size, this script has a parameter that allows you to optimize for either speed or space.

The syntax for nvTurboDatabase is:

```
nvTurboDatabase [ speed | space ]
```

Implementation

To implement database improvements, consider the following:

- "Improving Database Performance without NDBM Enhancements"
- "Migration Options" on page 92
- "Possible Migration Strategies" on page 92

Improving Database Performance without NDBM Enhancements

Before describing strategies for implementing the new NDBM enhancements, it is important to look at improving the operation of the key NDBM databases:

Table 12. Methods for Improving Database Performance

Database	Methods to Improve Performance
nodeinfo ifinfo	The netmon and the topology daemons generally update a node and interface twice during such operations as configuration checking. Buffering can reduce the number of disk writes by about half.
topoinfo netifno segnifo	The count fields are the most often updated fields in these databases. Buffering these fields with a small number of buffers will reduce unnecessary disk I/O.

Table 12. Methods for Improving Database Performance (continued)

Database	Methods to Improve Performance
obj_info	Hashing **-d o** will help group the data for one object into the same page. Buffering with a count of two should also be used here to fully realize the benefit of grouping the object data. Two problems are addressed with these changes: 1. The sparse file problems - the solution is to use an overflow file 2. Performance The default hashing algorithm places the different fields for any given object into several different areas (pages) of the **.pag** file. Performance will be greatly improved if the hashing is changed to **o**; **-h o** hashing tries to keep the fields for an object together in a small group of pages. When combined with buffering, this produces significant savings in ovwdb performance. The buffering count should be set to at least 3, (that is, **-b 3**).
name_inf	This database has a problem of storing different names for any given object in different pages. Hashing **-d n** will try to group the names for an object into the same page. This must also be used with buffering but a buffer value of one or two should be sufficient.
syminfo	This database will benefit from **-h o** hashing because symbols tend to be updated in groups with similar oids. Buffering with a count of two will also help this database run faster.
objinfo	Buffering will help this database to run faster

Migration Options

Following are migration strategies that you may want to pursue with regard to the NDBM enhancements:

1. Remaining with traditional NDBM processing

   No migration step are required.
2. Moving from traditional NDBM processing to enhanced NDBM processing

   This can be done on a database-by-database basis. For example, you could use traditional NDBM processing for all databases but for value_info, and use an overflow file for it. To migrate, run the dbmcompress program with new options set.
3. Moving from enhanced NDBM processing back to traditional NDBM processing

   This can be accomplished by running the dbmcompress program on a database with no options specified.

Possible Migration Strategies

Before implementing the NDBM enhancements in a production environment, ensure that there are procedures for regular backups in place. When making backup copies with NetView utilities, move the ***.BAK** files out of the **/usr/OV/databases** directory tree. Then, consider the following:

1. Minimizing space

   Run the **nvTurboDatabase space** script to limit sparse file system problems.
2. Maximizing performance (speed)

   Run the **nvTurboDatabase speed** script to maximize speed. This solution will work if your database is small enough or if your disk is large enough to allow the **value_info.pag** file to be backed up.
3. A combination of minimizing space and maximizing speed

For normal operation, use **nvTurboDatabase speed** to optimize speed. For backups, perform the following steps:

a. Run the **nvTurboDatabase space** script. This creates a database that minimizes space.
b. Run the **nvTurboDatabase speed** script. This creates a database that optimizes speed but also moves the database files above to new files with a **.BAK** extension.
c. Move the ***.BAK** files to a backup directory and then tar them up if necessary.

This strategy has the advantage of requiring less disk space and tarring to tapes more quickly. The disadvantage of this strategy is that the double compress takes longer to complete. Additionally, this approach may require that you write a small script to copy **.BAK** files to the backup directory and to tar them.

Appendix E. Files That Migrate

If you are migrating from Version 4, Version 5, or Version 5.1 or if you are reinstalling Version 7, you can migrate the following categories of files:

Table 13. File That Migrate

Directory	File Category
/usr/OV/ALL	All categories This includes all the categories listed in this section. Use this category if you want to migrate all data.
/usr/OV/ALL.USER	All user-defined categories This includes all the categories listed in this section except the categories that have the .USER extension, only the user-defined categories are migrated. For example, there are two categories for MIBs: **/usr/OV/snmp_mibs** and **/usr/OV/snmp_mibs.USER**. The .USER file contains the user-defined MIBs. If you select **/usr/OV/ALL.USER**, the **/usr/OV/snmp_mibs.USER** category is migrated, but the **/usr/OV/snmp_mibs** category is not.
/usr/OV/databases/openview	Topology map database This includes the ovwdb, mapdb, and topology databases. Server only.
/usr/OV/databases/snmpCollect	SNMP collection data This includes all data that the snmpCollect daemon gathers. The snmpCollect task definitions are stored in the /usr/OV/conf/snmpCol.conf file, which is migrated only if you select the /usr/OV/conf file category. Server only.
/usr/OV/servers/*Servername*/databases	Map database Client only.
/usr/OV/registration	Application registration files This includes all product-defined application registration files, user-added application registration files (ARFs), and all ARFs added by other integrated applications.
/usr/OV/fields	Field registration files This includes all product-defined and user-added field registration files (FRFs) and FRFs added by other integrated applications, except for the snmp_fields file. The snmp_fields file is not migrated. Server only.
/usr/OV/symbols	Symbol type registration files This includes all product-defined and user-added symbol type registration files (STRFs) and STRFs added by other integrated applications.
/usr/OV/lrf	Local registration files This includes all product-defined and user-added local registration files (LRFs) and LRFs added by other integrated applications. Server only.

Table 13. File That Migrate (continued)

Directory	File Category
Configuration files	/usr/OV/conf (Server only) This includes the following files: • HPoid2type • emstest.src • xmpcfg.dat • ovsuf • ovors • ovsnmp.conf • trapd.conf • oid_to_type • oid_to_protocol • oid_to_command • oid_to_label • mibExpr.conf • mib.coerce • mib.odi • mib2.def • snmpCol.conf • dbconf.dat • rdb_tracemask • ovevent.db • ovevent.dest • snmpColFiles • snmpmib • snmpmib.bin • nc.seed • “netmon seed file” • “backup manager seed file” • “server clients list”
Configuration files (continued)	/usr/OV/conf (Server only) (continued) This includes the following files: • “user-defined .modem files” • tralertd.conf • tralertd.default • tralertd.filter • tralertd_default.filter • nvdbtools/nvsniffer.conf • communityNames.conf • location.conf • mnpcodes.desc • mnpcodes.desc.undo • ESE.automation • nv.carriers • nvpaging.protocols • tecint.conf • nvpager.config • nvpager.warm • rulesets/Default.rs • rulesets/forwardall.rs • rulesets/* (user-added) • C/oid_to_sym • C/nnm_to_ovw • C/trapd.conf • C/if_to_sym

Table 13. File That Migrate (continued)

Directory	File Category
/usr/OV/app-defaults	Application default files This includes all product-defined X-Default files.
/usr/OV/security	Security files This includes all security configuration files, product-defined and user-added security registration files (SRFs), and SRFs added by other integrated applications. Server only.
/usr/OV/snmp_mibs	All loadable MIB files This includes all product-defined and user-added MIB files and MIBs added by other integrated applications. Server only.
/usr/OV/snmp_mibs.USER	User loadable MIB files This includes all the MIB files that were not originally installed with Tivoli NetView. This category is a subset of the /usr/OV/snmp_mibs category. Server only.
/usr/OV/reports	Report files This includes all product-defined and user-added reports and reports added by other integrated applications.
/usr/OV/filters	Filter files This includes all product-defined and user-added filters and filters added by other integrated applications.
/usr/OV/bitmaps	All bitmap files This includes all product-defined and user-added bitmaps and bitmaps added by other integrated applications.
/usr/OV/bitmaps.USER	User bitmap files This includes all the bitmap files that were not originally installed with Tivoli NetView. This category is a subset of the /usr/OV/bitmaps category.
/usr/OV/backgrounds	All background files This includes all product-defined and user-added backgrounds and backgrounds added by other integrated applications.
/usr/OV/backgrounds.USER	User background files This includes all the background files that were not originally installed with Tivoli NetView. This category is a subset of the /usr/OV/background category.
/usr/OV/icons	All icon files This includes all product-defined and user-added icon definition files and icon definition files added by other integrated applications.
/usr/OV/icons.USER	User icon files This includes all the icon definition files that were not originally installed with Tivoli NetView. This category is a subset of the /usr/OV/icons category.

Table 13. File That Migrate (continued)

Directory	File Category
/usr/OV/help	Help files This includes product-defined MIB application and user-added help files, and help files added by other integrated applications.
/usr/OV/cron	Cron files This includes all cron job scripts or cron job information. The active list of /usr/OV/crontab entries is saved in this directory.
/usr/OV/bin.USER	User bin files This includes all the scripts or executable files that were not originally installed with Tivoli NetView.

Appendix F. Additional Copyright and License Information

The product described in this document also contains software downloaded from several web servers. Permission to download and use such software is conditioned upon inclusion of the following notices.

gd 1.2 © Copyright 1994, 1995, Quest Protein Database Center, Cold Spring Harbor Labs. Permission granted to copy and distribute this work provided that this notice remains intact. Credit for the library must be given to the Quest Protein Database Center, Cold Spring Harbor Labs, in all derived works. This does not affect your ownership of the derived work itself, and the intent is to assure proper credit for Quest, not to interfere with your use of gd. If you have questions, ask. ("Derived works" includes all programs that utilize the library. Credit must be given in user-visible documentation.)

gd 1.2 was written by Thomas Boutell and is currently distributed by boutell.com, Inc.

If you wish to release modifications to gd, please clear them first by sending email to boutell@boutell.com; if this is not done, any modified version of the gd library must be clearly labeled as such.

The Quest Protein Database Center is funded under Grant P41-RR02188 by the National Institutes of Health.

Written by Thomas Boutell, 2/94–8/95.

The GIF compression code is based on that found in the pbmplus utilities, which in turn is based on GIFENCOD by David Rowley. See the notice below:

Based on GIFENCOD by David Rowley. A Lemple-Ziv compression based on "compress".

Modified by Marcel Wijkstra.

Copyright © 1989 by Jef Poskanzer.

Permission to use, copy, modify, and distribute this software and its documentation for any purpose and without fee is hereby granted, provided that the above copyright notice appears in all copies and that both that copyright notice and this permission appear in all supporting documentation. This software is provided "as is" without express or implied warranty.

The Graphics Interchange Format © is the Copyright property of CompuServe Incorporated. GIF (sm) is a Service Mark property of CompuServe Incorporated.

The GIF decompression is based on that found in the pbmplus utilities, which in turn is based on GIFDECOD by David Koblas. See the notice below:

Copyright 1990, 1991, 1993, David Koblas (koblas@netcom.com).

Permission to use, copy, modify, and distribute this software and its documentation for any purpose and without fee is hereby granted, provided that the above

copyright notice appears in all copies and that both that copyright notice and this permission appear in all supporting documentation. This software is provided “as is” without express or implied warranty.

GIFtrans v1.12

Convert any GIF file into a GIF89a. Allows for setting the transparent or background color, changing colors, adding or removing comments. Also code to analyze GIF contents.

Copyright © 24.2.94 by Andreas Ley

Permission to use, copy, modify, and distribute this software for any purpose and without fee is hereby granted, provided that the above copyright notice appears in all copies. This software is provided “as is” without express or implied warranties.

Glossary

This glossary defines technical terms used in the documentation for Tivoli products and includes selected terms and definitions from:

- The *American National Standard Dictionary for Information Systems* , ANSI X3.172-1990, copyright 1990 by the American National Standards Institute (ANSI). Copies may be purchased from the American National Standards Institute, 11 West 42nd Street, New York, New York 10036. Definitions are identified by the symbol (A) after the definition.
- The ANSI/EIA Standard—440-A, *Fiber Optic Terminology*. Copies may be purchased from the Electronic Industries Association, 2001 Pennsylvania Avenue, N.W., Washington, DC 20006. Definitions are identified by the symbol (E) after the definition.
- The *Information Technology Vocabulary* developed by Subcommittee 1, Joint Technical Committee 1, of the International Organization for Standardization and the International Electrotechnical Commission (ISO/IEC JTC1/SC1). Definitions of published parts of this vocabulary are identified by the symbol (I) after the definition; definitions taken from draft international standards, committee drafts, and working papers being developed by ISO/IEC JTC1/SC1 are identified by the symbol (T) after the definition, indicating that final agreement has not yet been reached among the participating National Bodies of SC1.
- The *IBM® Dictionary of Computing* , New York: McGraw-Hill, 1994.
- Internet Request for Comments: 1208, *Glossary of Networking Terms*
- Internet Request for Comments: 1392, *Internet Users' Glossary*
- The *Object-Oriented Interface Design: IBM Common User Access® Guidelines* , Carmel, Indiana: Que, 1992.

The following cross-references are used in this glossary:

Contrast with:
This refers the reader to a term that has an opposed or substantively different meaning.

See: This refers the reader to (a) a related term, (b) a term that is the expanded form of an abbreviation or acronym, or (c) a synonym or more preferred term.

Obsolete term for:
This indicates that the term should not be used and refers the reader to the preferred term.

A

ABAP/4. See Advanced Business Application Programming/4.

absolute path. A path that begins with the root directory. The absolute path may also be known as the "full pathname." Contrast with relative path.

abstract model. In Tivoli Global Enterprise Manager, the business description files that logically describe a particular business system.

abstract syntax notation 1 (ASN.1). The Open Systems Interconnection (OSI) method for abstract syntax specified in the following standards:

- ITU-T Recommendation X.208 (1988) | ISO/IEC 8824: 1990
- ITU-T Recommendation X.680 (1994) | ISO/IEC 8824-1: 1994

accelerator. (1) In a user interface, a key or combination of keys that invokes an application-defined function. (2) In the AIXwindows® Toolkit, a keyboard alternative to a mouse button action; for example, holding the <Shift> and <M> keys on the keyboard can be made to post a menu in the same way that a mouse button action does. Accelerators typically provide increased input speed and greater convenience.

access control. In computer security, the process of ensuring that the resources of a computer system can be accessed only by authorized users in authorized ways.

access control list. (1) In computer security, a collection of all access rights for one object. (2) In computer security, a list associated with an object that identifies all the subjects that can access the object and their access rights; for example, a list associated with a file that identifies users who can access the file and identifies their access rights to that file.

ACF. See Adapter Configuration Facility.

action. (1) An operation on a managed object, the semantics of which are defined as part of the managed object class definition. (2) In the AIX operating system,

a defined task that an application performs. An action modifies the properties of an object or manipulates the object in some way.

Action Message Retention Facility (AMRF). An OS/390 facility that, when active, retains all action messages except those specified by the installation.

adapter. (1) A part that electrically or physically connects a device to a computer or to another device. (2) Software that enables different software components or products to interact with one another. (3) See event adapter.

Adapter Configuration Facility (ACF). In the Tivoli Enterprise Console®, a graphical user interface that enables a Tivoli administrator to easily configure and customize event adapters.

ADE. See Tivoli Application Development Environment.

ADF. See application description file.

Administrative Domain. A collection of hosts and routers, and the interconnecting networks, managed by a single administrative authority.

administrator. See Tivoli administrator.

administrator collection. In a Tivoli environment, the collection for administrator objects that is generated by Tivoli Enterprise™ software. This container is represented by the Administrator icon on the Tivoli desktop; opening the icon provides access to information about each Tivoli administrator.

admin role. See authorization role.

Advanced Business Application Programming/4 (ABAP/4). A fourth-generation programming language in which SAP R/3 application software is written.

AEF. See Tivoli Application Extension Facility .

agent. (1) In systems management, a user that, for a particular interaction, has assumed an agent role. (2) An entity that represents one or more managed objects by (a) emitting notifications regarding the objects and (b) handling requests from managers for management operations to modify or query the objects. (3) A system that assumes an agent role.

Agent Policy Manager (APM). In Tivoli NetView, a function that controls Mid-Level Manager (MLM) configurations in a network from a single, central location.

agent role. In systems management, a role assumed by a user in which the user is capable of performing management operations on managed objects and of emitting notifications on behalf of managed objects.

aggregate object. In the NetView Graphic Monitor Facility, an object that represents a collection of real objects.

AIXwindows Toolkit. An object-oriented collection of C language data structures and subroutines that supplement the Enhanced X-Windows Toolkit and simplify the creation of interactive client application interfaces.

alarm. A signal, either audible or visual, at a device such as a display station or printer that is used to notify the user that a condition requiring the user's attention exists.

alarm level. In Tivoli Distributed Monitoring, the state of a monitor when a specified threshold has been reached. A Tivoli administrator can set thresholds for each alarm level and have Tivoli Distributed Monitoring trigger a different response (an action and an event) for each level. There can also be several responses for each alarm level.

alert. (1) A message sent to a management services focal point in a network to identify a problem or an impending problem. (2) In SNA management services (SNA/MS), a high priority event that warrants immediate attention.

alias name. A name that is defined in one network to represent a logical unit name in another interconnected network. The alias name does not have to be the same as the real name; if these names are not the same, translation is required.

alias name translation facility. In Tivoli NetView for OS/390, a function for converting logical unit names, logon mode table names, and class-of-service names used in one network into equivalent names to be used in another network.

allomorphism. The ability of an instance of a class to be managed as an instance of one or more different but compatible managed object classes.

AMP. See application management package.

AMRF. See Action Message Retention Facility.

AMS. See Application Management Specification.

AOF. See application object file.

AON. See Automated Operations Network.

APAR. See authorized program analysis report.

API. See application programming interface.

APM. See Agent Policy Manager.

application. A collection of software components used to perform specific types of user-oriented work on a computer.

application description file (ADF). In the context of the Application Management Specification (AMS), a readable, ASCII text file that contains information for managing an application. Application description files are based on the Management Information Format (MIF). Application description files include component description files, global description files, and business description files (business system description files, business system component description files, business system mapping description files, and business subsystem description files).

application management package (AMP). In a Tivoli environment, a compressed file that contains the application description files and other necessary files for managing an application. These include one global description file, one or more component description files, task scripts, and executable programs. The application management package can also include the application object file or the source files for the application itself.

Application Management Specification (AMS). A specification that presents a standard for managing applications. The Application Management Specification was developed in collaboration with the Tivoli Partners and Tivoli customers to address the problems associated with multitiered applications.

application object file (AOF). In a Tivoli environment, an ASCII text file that contains the names of the global description file and the component description files, which together describe the management characteristics of an application. The Tivoli Module Designer and the Tivoli Module Builder can import an application object file that was created by the obsolete Tivoli Developer Kit.

application plane. In Tivoli NetView, the submap layer on which symbols of objects that are managed by at least one network or systems management application program are displayed. Symbols on the application plane are displayed without shading, which makes them appear directly against the background plane. See user plane.

application programming interface (API). A software interface that enables applications to communicate with each other. An API is the set of programming language constructs or statements that can be coded in an application program to obtain the specific functions and services provided by an underlying operating system or service program.

application registration file (ARF). A file created to integrate an application program into Tivoli NetView by defining (a) the application program's position in the menu structure for Tivoli NetView, (b) where help information is found, (c) the number and types of parameters allowed, (d) the command used to start the application program, and (e) other characteristics of the application program.

Application Response Measurement (ARM). An application programming interface that was developed by a group of leading technology vendors, including Tivoli Systems Inc., and that can be used to monitor the availability and performance of business transactions within and across diverse applications and systems. The monitoring is done from the perspective of the applications; therefore, it reflects the units of work that are important from the perspective of the business. For example, using ARM, a business could instrument an application to discover:

- Whether the application is hung
- The level of response time that the application is experiencing
- Where the bottlenecks are occurring during the execution of the application
- Who is using the application and how much they are using it
- How to tune the system environment to run the application more efficiently
- What the application is doing during the reported response time
- Where in the system environment a transaction is spending its time

APPNTAM. See SNA topology manager.

APPN® Topology and Accounting Manager (APPNTAM). See SNA topology manager.

ARF. See application registration file.

ARM. (1) See Application Response Measurement. (2) See automatic restart manager.

ARM agent. An agent that monitors software that is instrumented using the Application Response Measurement (ARM). The ARM agent is shipped as part of Tivoli Distributed Monitoring.

ASN.1. See abstract syntax notation 1.

ASYNC. See asynchronous.

asynchronous (ASYNC). (1) Pertaining to two or more processes that do not depend upon the occurrence of specific events such as common timing signals. (T) (2) Without regular time relationship; unexpected or unpredictable with respect to the execution of program instructions.

asynchronous monitor. In Tivoli Distributed Monitoring, a monitor that receives data in an unsolicited event and interprets the data immediately. Contrast with synchronous monitor.

attribute. A characteristic that identifies and describes a managed object. The characteristic can be determined, and possibly changed, through operations on the managed object.

authentication. (1) In computer security, verification of the identity of a user or the user's eligibility to access an object. (2) In computer security, verification that a message has not been altered or corrupted. (3) In computer security, a process used to verify the user of an information system or protected resources.

authorization. (1) In computer security, the right granted to a user to communicate with or make use of a computer system. (T) (2) An access right. (3) The process of granting a user either complete or restricted access to an object, resource, or function.

authorization role. In a Tivoli environment, a role assigned to Tivoli administrators to enable them to perform their assigned systems management tasks. A role may be granted over the entire Tivoli Management Region or over a specific set of resources, such as those contained in a policy region. Examples of authorization roles include: super, senior, admin, and user.

authorized operator. In Tivoli NetView for OS/390, an operator who has been authorized to receive undeliverable messages and lost terminal messages. See authorized receiver.

authorized program analysis report (APAR). A report of a problem caused by a suspected defect in a current unaltered release of a program.

authorized receiver. In Tivoli NetView for OS/390, an authorized operator who receives the unsolicited and authorized-receiver messages that are not assigned to a specific operator.

Automated Operations Network (AON). In Tivoli NetView for OS/390, the component that handles automated resource monitoring, recovery, and tracking.

automated response. In a Tivoli environment, a predefined response to particular events that is automatically executed by a Tivoli application. For example, if the Tivoli Enterprise Console detects that a process has terminated early, it can automatically restart the process without the intervention of the Tivoli administrator.

automatic reactivation. In Tivoli NetView for OS/390, the activation of a node from the inactive state without any action by the network operator.

automatic restart manager (ARM). An OS/390 recovery function that can automatically restart batch jobs and started tasks after they or the system on which they are running terminate unexpectedly.

AutoPack Control Center. A Tivoli Software Distribution tool that is installed on a Windows-based PC and enables a Tivoli administrator to create an AutoPack file. The AutoPack Control Center produces the AutoPack file by (a) taking snapshots of the PC's drive and system configuration before and after the installation of an application on the PC and (b) capturing the differences between these snapshots and the distribution instructions in the AutoPack file.

AutoPack file. In Tivoli Software Distribution, an installable image that is used to distribute "shrinkwrapped" applications to multiple PC targets. The file contains a description of PC software application files and directories, information on how to distribute these files and directories, and any system configuration changes needed by the application. A Tivoli administrator must associate an AutoPack file with an AutoPack profile.

AutoPack profile. A Tivoli Software Distribution profile that references an AutoPack file.

autotask. (1) In Tivoli NetView for OS/390, an unattended operator station task that does not require a terminal or a logged-on user. Autotasks can run independently of VTAM® and are typically used for automated console operations. (2) Contrast with logged-on operator.

availability management. The Tivoli management discipline that addresses the gathering, collecting, and routing of information regarding the operational status of an organization's network computing system and enables the appropriate corrective action. See deployment management, operations and administration, and security management.

B

backend. In the AIX operating system, the program that sends output to a particular device.

background plane. In Tivoli NetView, the lowest submap layer. The background plane provides the background against which symbols are displayed. A background picture can be placed in the background plane to provide a context for viewing symbols. See application plane and user plane.

background process. (1) A process that does not require operator intervention but can be run by the computer while the workstation is used to do other work. (2) In the AIX operating system, a mode of program execution in which the shell does not wait for program completion before prompting the user for another command. (3) Contrast with foreground process.

background task. A task that is running even though the user is not currently interacting with it. Contrast with foreground task.

bandwidth. A measure of the capacity of a communication transport medium (such as a TV cable) to convey data.

BARC program. Obsolete term for configuration program. “BARC” is an acronym for “before, after, removal, and commit.”

BAROC. See Basic Recorder of Objects in C.

base module. In a Tivoli environment, a management module that describes the basic management characteristics of a particular application or business system to the Tivoli management software. Unlike Tivoli GEM modules and Tivoli Plus modules, base modules are developed without the use of a template.

bash. Bourne-again shell. A portable, command-line interface and script interpreter that is compatible with the UNIX Bourne and Korn shells and includes some features of the UNIX C shell.

Basic Input/Output System (BIOS). Code that controls basic hardware operations, such as interactions with diskette drives, hard disk drives, and the keyboard.

Basic Object Adapter (BOA). Software that provides CORBA-compliant services for object implementations.

Basic Recorder of Objects in C (BAROC). In the event server of the Tivoli Enterprise Console, the internal representation of the defined event classes.

basic sequential access method (BSAM). In the NetView Performance Monitor (NPM), the method by which all PIUs collected for selected LUs can be logged into a sequential data set as they pass through VTAM.

BCDF. See business system component description file.

BDF. See business description file.

bean. A reusable Java™ component that is built using the JavaBeans™ technology.

bilingual command list. In Tivoli NetView for OS/390, a command list written in a combination of REXX and the NetView command list language.

BIOS. (1) See Basic Input/Output System. (2) See NetBIOS.

bitmap. (1) A representation of an image by an array of bits. (2) A pixmap with a depth of one bit plane.

BMDF. See business system mapping description file.

BOA. See Basic Object Adapter.

bridge. (1) A functional unit that interconnects two local area networks that use the same logical link control protocol but may use different medium access control protocols. (T) (2) A functional unit that interconnects multiple LANs (locally or remotely) that use the same logical link control protocol but that can use different medium access control protocols. A bridge forwards a frame to another bridge based on the medium access control (MAC) address. (3) In the connection of local loops, channels, or rings, the equipment and techniques used to match circuits and to facilitate accurate data transmission. (4) Contrast with gateway and router.

browse. (1) To look at records in a file. (2) In the NetView Graphic Monitor Facility, to open a view that cannot receive status changes from Tivoli NetView for OS/390. Contrast with monitor.

BSAM. See basic sequential access method.

BSDF. See business system description file.

BSSDF. See business subsystem description file.

buffer. (1) A routine or storage used to compensate for a difference in rate of flow of data, or time of occurrence of events, when transferring data from one device to another. (A) (2) To allocate and schedule the use of buffers. (A) (3) A portion of storage used to hold input or output data temporarily.

bulletin board. In the Tivoli environment, the primary mechanism by which the Tivoli Management Framework and Tivoli applications communicate with Tivoli administrators. The bulletin board is represented as an icon on the Tivoli desktop through which the administrators can access notices. Tivoli applications use the bulletin board as an audit trail for important operations that the administrators perform.

business component. An application or other system resource that can be managed by systems management software.

business description file (BDF). In a Tivoli environment, a generic name for any of these application description files: business system description file (BSDF), business system component description file (BCDF), and business system mapping description file (BMDF), and business subsystem description file (BSSDF).

business subsystem description file (BSSDF). In the context of the Application Management Specification (AMS), an optional application description file that enables the logical grouping of business components in a business system. In this file, a Tivoli administrator can specify tasks and monitors that are common to the subsystem. The business subsystem description file references the applicable business system description file and one or more business system component description files.

business system. A group of diverse but interdependent applications and other system resources that interact to accomplish specific business functions.

business system component description file (BCDF). In the context of the Application Management Specification (AMS), an application description file that

defines the logical representation of a business component. When defining a business system, at least one business system component description file is required for each business system definition.

business system description file (BSDF). In the context of the Application Management Specification (AMS), the highest-level application description file that identifies the components of a business system, including monitors, tasks, and connections. A Tivoli administrator can also define icon and help files at this level (for creating a business system icon and help information).

business system mapping description file (BMDF). In the context of the Application Management Specification (AMS), the application description file that maps a real business component (which is defined in a component description file) to a logical business component (which is represented in a business system component description file). Each business system mapping description file references a business system component description file.

C

CADAM. Computer-Aided Design and Manufacturing. The use of computers in the design and manufacture of products such as cars, airplanes, ships, and computers.

call. (1) The action of bringing a computer program, a routine, or a subroutine into effect, usually by specifying the entry conditions and jumping to an entry point. (I) (A) (2) In data communication, the actions necessary to make a connection between two stations on a switched line. (3) In communications, a conversation between two users. (4) To transfer control to a procedure, program, routine, or subroutine. (5) To attempt to contact a user, regardless of whether the attempt is successful.

callback. In the AIX operating system, a procedure that is called if and when certain specified conditions are met.

canonical. In computer science, pertaining to an expression that conforms to a specific set of rules.

CATIA. Computer-Graphics Aided Three-Dimensional Interactive Application.

CC. See change control.

CCMS. (1) See Computing Center Management System. (2) See Configuration Change Management System.

CDF. See component description file.

CDNM session. See cross-domain network manager session.

CDS. See control data set.

central site control facility (CSCF). In Tivoli NetView for OS/390, NetView for VM, and NetView for VSE, a function that allows a network operator to execute the test facilities of the IBM 3172 Nways® Interconnect Controller and the IBM 3174 Establishment Controller remotely from the NetView console.

change control (CC). The use of change management commands for the installation or removal of software or data.

change control administrator. A person responsible for software distribution and change control activities.

change control client. A workstation that (a) receives software and data files from its change control server and (b) installs and removes software and data files as instructed by its change control server.

change control domain. A change control server and its change control clients.

change control server. A workstation that controls and tracks the distribution of software and data files to other workstations.

change control single node. A workstation that controls, tracks, installs, and removes software and data files for itself. A CC single node can also prepare software for distribution. Contrast with change control client and change control server.

change management. The process of planning (for example, scheduling) and controlling (for example, distributing, installing, and tracking) software changes over a network. This is sometimes known as “software management.”

check box. A square box with associated text that represents a choice. When a user selects the choice, the check box is filled to indicate that the choice is selected. The user can clear the check box by selecting the choice again, thereby deselecting the choice.

checkpoint. (1) Information about the status of a program’s execution or the status of a data transfer that is recorded to enable the program or the data transfer to be restarted if it is ever interrupted. (2) The time at which such information is recorded. (3) To record such information.

child process. In the UNIX operating system, a process, started by a parent process, that shares the resources of the parent process. See fork.

child resource. In the NetView Graphic Monitor Facility, a resource that is directly subordinate to another resource (the parent) in a hierarchy.

CICS®. See Customer Information Control System.

class. (1) In object-oriented design or programming, a model or template that can be instantiated to create

objects with a common definition and therefore, common properties, operations, and behavior. An object is an instance of a class. (2) In the AIX operating system, pertaining to the I/O characteristics of a device. System devices are classified as block or character devices.

CLI. See command line interface.

client. A computer system or process that requests a service of another computer system or process that is typically referred to as a server. Multiple clients may share access to a common server.

client daemon. An AIX process that performs the client's operations.

client/server. In communications, the model of interaction in distributed data processing in which a program at one site sends a request to a program at another site and awaits a response. The requesting program is called a client; the answering program is called a server.

client workstation. In the NetView Graphic Monitor Facility, a workstation that depends on a server workstation to provide it with views and status information. A client workstation receives status information from the server workstation over an LU 6.2 session.

cloning. (1) In a Tivoli environment, an operation that enables a Tivoli administrator to replicate profiles. This capability simplifies the task of creating multiple profiles with similar properties. See prototype profile. (2) In a Tivoli environment, a function of Tivoli NetView for OS/390 that enables a system programmer to replicate NetView definitions across the systems comprising a sysplex, thus simplifying the task of creating multiple NetView definitions with similar properties.

CNM. See communication network management.

CNM application program. A VTAM application program that issues and receives formatted management services request units for physical units. Tivoli NetView for OS/390 is an example of a CNM application program.

CNM processor. In Tivoli NetView for OS/390, a program that manages one of the functions of a communication system. A CNM processor is executed under control of Tivoli NetView for OS/390.

collaborative management. A cooperative relationship between Internet commerce partners and Internet service providers (ISPs) to ensure the successful completion of business transactions.

collection. In a Tivoli environment, a container that groups objects on a Tivoli desktop, thus providing the Tivoli administrator with a single view of related resources. Either the Tivoli Management Framework or a Tivoli administrator can create a collection. The contents of a collection are referred to as its members. Examples of collections include the administrator collection and the generic collection; the administrator collection is an example of a collection generated by the Tivoli Management Framework.

collection point block (CPB). In the NetView Performance Monitor (NPM), a control block used to coordinate the collection of network and session data.

combined alert. In Tivoli NetView for OS/390, an alert that includes elements of a non-generic and a generic alert in one network management vector transport (NMVT).

command. (1) A request from a terminal for the performance of an operation or the execution of a particular program. (2) In Tivoli NetView for OS/390, a sequence of characters that is submitted to cause an action. A command contains a verb and an object.

command authorization. The process of authorizing a network operator to use various commands. See NetView command authorization table, Resource Access Control Facility, scope of command authorization, and System Authorization Facility.

command facility. In Tivoli NetView for OS/390, the component that is a base for command processors that can monitor, control, automate, and improve the operation of a network.

command indicator. In the NetView Graphic Monitor Facility, a numeric identifier that is assigned to a network resource by its controlling resource manager to indicate the command support characteristics for the resource.

command interpreter. In the AIX operating system, a program that sends instructions to the kernel.

command line interface (CLI). A type of computer interface in which the input command is a string of text characters. Contrast with graphical user interface.

command list. In Tivoli NetView for OS/390, a list of commands and statements designed to perform a specific function for the user. Command lists can be written in REXX or in the NetView command list language.

command procedure. In Tivoli NetView for OS/390, a command list, a command processor written in a high-level language (HLL), or a NetView pipeline.

command processor. In Tivoli NetView for OS/390, a module designed to perform a specific function for the user. Users can write command processors in assembler language or in a high-level language (HLL); command processors are invoked as commands.

command profile editor (CPE). In Tivoli Global Enterprise Manager and Tivoli NetView for OS/390, a function of the topology console that enables Tivoli administrators who have the proper administrative authority to control the content, order, and capabilities of pop-up menus for individual operators or groups of operators.

commit operation. In Tivoli Software Distribution, an operation performed by a configuration program on target managed nodes after a file package distribution. This function enables a Tivoli administrator to distribute a file package to multiple targets and to make the distributed information available on all targets at the same time.

Common Object Request Broker Architecture (CORBA). A specification produced by the Object Management Group (OMG) that presents standards for various types of object request brokers (such as client-resident ORBs, server-based ORBs, system-based ORBs, and library-based ORBs). Implementation of CORBA standards enables object request brokers from different software vendors to interoperate.

Common Programming Interface for Communications (CPI-C). An evolving application programming interface (API), embracing functions to meet the growing demands from different application environments and to achieve openness as an industry standard for communications programming. CPI-C provides access to interprogram services such as (a) sending and receiving data, (b) synchronizing processing between programs, and (c) notifying a partner of errors in the communication.

communication network management (CNM). The process of designing, installing, operating, and managing distribution of information and control among users of communication systems.

communications infrastructure. In the AIX operating system, a framework of communication that consists of a postmaster, an object registration service, a startup file, communication protocols, and application programming interfaces.

Communications Server. An IBM licensed program that supports (a) the development and use of application programs across two or more connected systems or workstations, (b) multiple concurrent connections that use a wide range of protocols, and (c) several application programming interfaces (APIs) that may be called concurrently and that are designed for client/server and distributed application programs. Communications Server includes the necessary interfaces for network management and is available on several operating systems (such as AIX, OS/2® Warp, OS/390, and Windows NT®).

community. In the Simple Network Management Protocol (SNMP), an administrative relationship between entities.

community name. In the Simple Network Management Protocol (SNMP), a string of octets identifying a community.

component description file (CDF). In the context of the Application Management Specification (AMS), an application description file that contains information about a specific component in a management-ready application. Each management-ready application can contain multiple components, each of which is represented by one component description file.

Computing Center Management System (CCMS). The SAP interface for monitoring a SAP R/3 system.

configuration. (1) The manner in which the hardware and software of an information processing system are organized and interconnected. (T) (2) The devices and programs that make up a system, subsystem, or network.

Configuration Application. See MLM Configuration Application.

Configuration Change Management System (CCMS). In a Tivoli environment, a distributed, hierarchical database in which configuration data is stored for use by systems management applications in effecting configuration changes on groups of systems.

configuration file. A file that specifies the characteristics of a system device or network.

configuration management. The control of information necessary to identify both physical and logical information system resources and their relationship to one another.

configuration program. In Tivoli Software Distribution, a feature that enables a Tivoli administrator to perform operations (a) before or after file package distributions, (b) before or after file package removal, (c) during a file package commit operation, or (d) after an error stops a distribution or removal operation.

configuration repository. In a Tivoli environment, the relational database that contains information that is collected or generated by Tivoli applications. Following are examples of the information that is stored in the configuration repository:

- Tivoli Enterprise Console stores information regarding events.
- Tivoli Inventory stores information regarding hardware, software, system configuration, and physical inventory.
- Tivoli Software Distribution stores information regarding file package operations.

connector class. In Tivoli NetView, an object class used for objects that connect different parts of the network and that route or switch traffic between these parts. This class includes gateways, repeaters (including multiport repeaters), and bridges. Contrast with network class.

console event. In a Tivoli environment, an event sent to the Tivoli Enterprise Console.

container. A visual user-interface component that holds objects.

control data set (CDS). In the NetView Performance Monitor (NPM), a System Modification Program (SMP) data set used in the NPM installation process.

control desk. In Tivoli NetView, a component of the graphical user interface (GUI) that enables the network operator to group application program instances together.

control program. (1) A computer program designed to schedule and to supervise the execution of programs of a computer system. (I) (A) (2) The part of the AIX operating system that determines the order in which basic functions should be performed.

control statement. In Tivoli NetView for OS/390, a statement in a command list that controls the processing sequence of the command list or allows the command list to send messages to the operator and receive input from the operator.

CORBA. See Common Object Request Broker Architecture.

correlation activity. See event correlation.

CPB. See collection point block.

CPE. See command profile editor.

CPI-C. See Common Programming Interface for Communications.

critical resource. In the NetView Graphic Monitor Facility, a resource that is considered important to the operation of the network and therefore has a high aggregation priority.

cron table. In the AIX operating system, a table that is used to schedule application programs and processes. “Cron” is an abbreviation for “chronological.”

cross-domain network manager session. A session between two network managers (for example, Tivoli NetView for OS/390) in separate domains.

cross-system coupling facility (XCF). A component of the MVS™ operating system that provides functions to support cooperation between authorized programs running within a sysplex.

CSCF. See central site control facility.

current directory. See working directory.

Customer Information Control System (CICS). An IBM licensed program that provides online transaction processing services and management for critical business applications. CICS runs on many IBM and non-IBM platforms (from the desktop to the mainframe) and is used in various types of networks that range in size from a few terminals to many thousands of terminals. The CICS application programming interface (API) enables programmers to port applications among the hardware and software platforms on which CICS is available. Each product in the CICS family can interface with the other products in the CICS family, thus enabling interproduct communication.

custom monitor. In Tivoli Distributed Monitoring, a monitor that is implemented as a script or program by the Tivoli administrator.

D

daemon. A program that runs unattended to perform a standard service. Some daemons are triggered automatically to perform their task; others operate periodically.

DASD conservation option. In Tivoli NetView for OS/390, an installation option that allows Tivoli NetView for OS/390 to be installed without the online help facility and hardware monitor data presentation panels.

database. (1) A collection of data with a given structure for accepting, storing, and providing, on demand, data for multiple users. (T) (2) A collection of interrelated data organized according to a database schema to serve one or more applications. (T) (3) A collection of data fundamental to a system. (A) (4) A collection of data fundamental to an enterprise. (A)

data model. (1) A logical view of the organization of data in a database. (T) (2) In a database, the user’s logical view of the data in contrast to the physically stored data, or storage structure. (A) (3) A description of the organization of data in a manner that reflects the information structure of an enterprise. (A)

data modeling. A structured set of techniques for defining and recording business information requirements. It is a depiction of the user’s view of the data needs of the organization in a consistent and rigorous fashion. The data model eventually serves as the basis for translation to computer system databases.

data services command processor (DSCP). In Tivoli NetView for OS/390, a component that structures a request for recording and retrieving data in the application program’s database and for soliciting data from a device in the network.

data services manager (DSM). In Tivoli NetView for OS/390, a function that provides VSAM services for data storage and retrieval.

data services request block (DSRB). In Tivoli NetView for OS/390, the control block that contains information that a data services command processor (DSCP) needs to communicate with the data services task (DST).

data services task (DST). In Tivoli NetView for OS/390, the subtask that gathers, records, and manages data in a VSAM file or a network device that contains network management information.

data type. In Tivoli NetView for OS/390, one of the three elements, which also include display type and resource type, that are used to describe the organization of panels. Data types include alerts, events, and statistics.

dce-pipe-pull. A Printing Systems Manager (PSM) document transfer method in which the client saves documents in a file and transfers the address of the file to the server. The file is later transferred to the server upon request from the server. This is an efficient transfer method for large jobs. Contrast with with-request.

default policy. In a Tivoli environment, a set of resource property values that are assigned to a resource when the resource is created.

definition statement. (1) In VTAM, the statement that describes an element of the network. (2) In NCP, a type of instruction that defines a resource to the NCP.

defragmentation. The process of running a software utility to rewrite fragmented data to contiguous sectors of a computer storage medium to improve access and retrieval time. Contrast with fragmentation.

demand poll. In Tivoli NetView, a polling operation initiated by the user.

deployment management. The Tivoli management discipline that addresses the automation of configuration and change management activities for the ever-evolving components of a network computing system. See availability management, operations and administration, and security management.

desktop. See Tivoli desktop.

Desktop Management Interface (DMI). A protocol-independent set of application programming interfaces (APIs) defined by the Desktop Management Task Force (DMTF). These interfaces give management application programs standardized access to information about hardware and software in a system.

Desktop Management Task Force (DMTF). An alliance of computer vendors that was convened to define streamlined management of the diverse operating systems commonly found in an enterprise.

developer key. In the context of SAP application software, a key that is provided by SAP for a developer's use in creating or changing Advanced Business Application Programming (ABAP) objects.

DFSMSdfp™. A DFSMS/MVS® component and a base element of OS/390 that provides functions for storage management, data management, program management, device management, and distributed data access (″dfp″ represents ″data facility product″).

DFSMSdss™. A DFSMS/MVS component and a base element of OS/390 that is used in copying, moving, dumping, defragmenting, and restoring data sets and volumes (″dss″ represents ″data set services″).

DFSMShsm™. A DFSMS/MVS component and a base element of OS/390 that is used in backing up data, in recovering data, in managing storage space on volumes in the storage hierarchy, and in disaster recovery (″hsm″ represents ″hierarchical storage manager″).

DFSMS/MVS. An IBM licensed program that provides storage, data, and device management functions in an MVS/ESA™ Version 5 or an OS/390 environment. DFSMS/MVS includes these components: DFSMSdfp, DFSMSdss, DFSMShsm, and DFSMSrmm™. ″DFSMS″ represents ″Data Facility Storage Management Subsystem.″

DFSMSrmm. A DFSMS/MVS component and base element of OS/390 that manages removable media (″rmm″ represents ″removable media manager″).

DHCP. See Dynamic Host Configuration Protocol.

directory. In a hierarchical file system, a container for files or other directories. See path.

discriminator. An object that enables a system to select operations and event reports relating to other managed objects. See event forwarding discriminator.

display type. In Tivoli NetView for OS/390, one of the three elements, which also include data type and resource type, that are used to describe the organization of panels. Display types include total, most recent, user action, and detail.

distributed computing. See network computing.

Distributed Monitoring engine. In a Tivoli environment, the client software that is installed on each managed node, gateway, and endpoint that is being monitored by Tivoli Distributed Monitoring. The Distributed Monitoring engine monitors resources, compares data from monitored resources against configured thresholds, and runs automated responses.

Distributed Monitoring proxy. See endpoint.

distribution program. See configuration program.

DMI. See Desktop Management Interface.

DMTF. See Desktop Management Task Force.

domain. (1) That part of a computer network in which the data processing resources are under common control. (T) (2) See Administrative Domain and domain name.

domain name. In the Internet suite of protocols, a name of a host system. A domain name consists of a sequence of subnames separated by a delimiter character. For example, if the fully qualified domain name (FQDN) of a host system is `ralvm7.vnet.ibm.com` , each of the following is a domain name:

- `ralvm7.vnet.ibm.com`
- `vnet.ibm.com`
- `ibm.com`

double recording. In Tivoli NetView for OS/390, pertaining to the recording of certain individual events under two resource levels.

downcall. In a Tivoli environment, a method invocation from the TMR server or the gateway “down” to an endpoint. Contrast with upcall.

drag and drop. To directly manipulate an object by moving it and placing it somewhere else using a pointing device (such as a mouse).

DSCP. See data services command processor.

DSM. See data services manager.

DSRB. See data services request block.

DST. See data services task.

Dynamic Host Configuration Protocol (DHCP). A protocol defined by the Internet Engineering Task Force (IETF) that is used for dynamically assigning IP addresses to computers in a network.

E

e-business. Either (a) the transaction of business over an electronic medium such as the Internet or (b) any organization (for example, commercial, industrial, nonprofit, educational, or governmental) that transacts its business over an electronic medium such as the Internet. An e-business combines the resources of traditional information systems with the vast reach of an electronic medium such as the Internet (including the World Wide Web, intranets, and extranets); it connects critical business systems directly to critical business constituencies--customers, employees, and suppliers. The key to becoming an e-business is building a transaction-based Web site in which all core business processes (especially all processes that require a dynamic and interactive flow of information) are put online to improve service, cut costs, and sell products.

ECB. See event control block.

e-commerce. The subset of e-business that involves the exchange of money for goods or services purchased over an electronic medium such as the Internet.

EFD. See event forwarding discriminator.

EIF. See Tivoli Event Integration Facility.

EMS. See event management services.

encapsulation. (1) In object-oriented programming, the technique that is used to hide the inherent details of an object. This technique is also known as “information hiding.” (2) In object-oriented programming, a software technique in which data is packaged with corresponding procedures. In CORBA, the object is the mechanism for encapsulation.

endpoint. (1) In a Tivoli environment, a Tivoli client that is the ultimate recipient for any type of Tivoli operation. (2) In a Tivoli environment, a Tivoli service that runs on multiple operating systems and performs Tivoli operations on those systems, thereby enabling the Tivoli Management Framework to manage the systems as Tivoli clients.

endpoint list. In a Tivoli environment, a list of all endpoint clients in the Tivoli Management Region with their assigned gateways. See endpoint manager.

endpoint manager. In a Tivoli environment, a service that runs on the Tivoli server, assigns endpoint clients to gateways, and maintains the endpoint list.

endpoint method. In a Tivoli environment, a method that runs on an endpoint client as the result of a request from other managed resources in the Tivoli Management Region. Results of the method are forwarded first to the gateway, then to the calling managed resource.

Enhanced X-Windows Toolkit. (1) In the AIX operating system, a collection of basic functions for developing a variety of application environments. Toolkit functions manage Toolkit initialization, widgets, memory, events, geometry, input focus, selections, resources, translation of events, graphics contexts, pixmaps, and errors. (2) See AIXwindows Toolkit and X Window System.

entity. Any concrete or abstract thing of interest, including associations among things; for example, a person, object, event, or process that is of interest in the context under consideration, and about which data may be stored in a database. (T)

entry point (EP). (1) The address or label of the first instruction executed on entering a computer program,

routine, or subroutine. A computer program, routine, or subroutine may have a number of different entry points, each perhaps corresponding to a different function or purpose. (I) (A) (2) In SNA, a type 2.0, type 2.1, type 4, or type 5 node that provides distributed network management support. It sends network management data about itself and the resources it controls to a focal point for centralized processing, and it receives and executes focal-point initiated commands to manage and control its resources.

EP. See entry point.

error record template. In the AIX operating system, a template that describes the error class, error type, error description, probable causes, recommended actions, and failure data for an error log entry.

euro. The monetary unit of the European Monetary Union (EMU) that will be introduced alongside national currencies on the first of January 1999. In May 1998, eleven countries were confirmed for EMU membership beginning the first of January 1999: Austria, Belgium, Finland, France, Germany, Ireland, Italy, Luxembourg, the Netherlands, Portugal, and Spain. On the first of January 2002, euro notes and coins (hard currency) will be put into circulation, and national currencies will be withdrawn, probably over a six-month period.

EuroReady product. A product is EuroReady if the product, when used in accordance with its associated documentation, is capable of correctly processing monetary data in the euro denomination, respecting the euro currency formatting conventions (including the euro sign). This assumes that all other products (for example, hardware, software, and firmware) that are used with this product are also EuroReady. IBM hardware products that are EuroReady may or may not have an engraved euro sign key on their keyboards.

EuroReady solution. A solution is EuroReady when the solution providers have done the following:

1. Analyzed the euro requirements, including the need to comply with relevant European Community (EC) rules
2. Included the appropriate function according to these requirements
3. Clearly demonstrated this by (a) detailing the euro-related requirements, (b) describing how these requirements will be implemented, and (c) declaring when the implementation will be generally available.

event. (1) An occurrence of significance to a task (such as the opening of a window or the completion of an asynchronous operation). (2) In the Tivoli environment, any significant change in the state of a system resource, network resource, or network application. An event can be generated for a problem, for the resolution of a problem, or for the successful completion of a task. Examples of events are: the normal starting and stopping of a process, the abnormal termination of a process, and the malfunctioning of a server. (3) See event report.

event adapter. In a Tivoli environment, software that converts events into a format that the Tivoli Enterprise Console can use and forwards the events to the event server. Using the Tivoli Event Integration Facility, an organization can develop its own event adapters, tailored to its network environment and specific needs.

event/automation service. In Tivoli NetView for OS/390, a facility that translates alerts and messages into events for the Tivoli Enterprise Console and translates these events into NetView alerts. The event/automation service communicates with Tivoli NetView for OS/390 using the program-to-program interface (PPI), and it communicates with the Tivoli Enterprise Console using TCP/IP.

event card. In Tivoli NetView, a graphical representation, resembling a card, of the information contained in an event sent by an agent to a manager reflecting a change in the status of one of the agent's managed nodes.

event class. In the Tivoli Enterprise Console, a classification for an event that indicates the type of information that the event adapter will send to the event server.

event console. In the Tivoli Enterprise Console, a graphical user interface (GUI) that enables system administrators to view and respond to dispatched events from the event server. The Tivoli Event Integration Facility does not directly use or affect event consoles.

event control block (ECB). A control block used to represent the status of an event.

event correlation. In the Tivoli Enterprise Console, the process of correlating separate events to a common cause. For example, the Tivoli Enterprise Console may receive several `NFS server not responding` events from several different applications, as well as a host down event for the NFS server. The Tivoli Enterprise Console can then correlate the various `NFS server not responding` events to their common cause, which is: the NFS server is "down." See rule.

event filter. (1) In a Tivoli environment, software that determines which events are forwarded to a specified destination. Filtering events helps to reduce network traffic. Tivoli administrators configure the event filters. (2) In Tivoli NetView, a logical expression of criteria that determine which events are forwarded to the application program that registers the event filter with the event sieve agent. A filter is referred to as "simple" or "compound" depending on how it is handled by the filter editor.

event forwarding discriminator (EFD). A managed object that describes the criteria used to select which event reports are sent and to whom they are sent.

event group. In the Tivoli Enterprise Console, a set of events that meet certain criteria. Each event group is represented by an icon on the event console. Tivoli administrators can monitor event groups that are relevant to their specific areas of responsibility.

event handler. A collection and correlation point for events and messages.

event management services (EMS). In Tivoli NetView, a centralized method of generating, receiving, routing, and logging network events.

event manager. In the NetView Graphic Monitor Facility, the component of the host subsystem that receives alert and resolution major vectors from Tivoli NetView for OS/390, translates these major vectors into generic event records, and applies the event status to the resource defined in the Resource Object Data Manager (RODM) cache.

event report. The unsolicited report that an event has occurred. In an Open Systems Interconnection (OSI) context, when a managed object emits a notification, the agent uses one or more event forwarding discriminators (EFDs) to find the destinations to which the report is sent.

event repository. See configuration repository.

event server. In the Tivoli Enterprise Console, a central server that processes events. The event server creates an entry for each incoming event and evaluates the event against a rule base to determine whether it can respond to or modify the event automatically. The event server also updates the event consoles with the current event information. If the primary event server is not available, events can be sent to a secondary event server.

event sieve. In Tivoli NetView, an object that is managed by the “ovesmd” daemon, which is the event sieve agent. The event sieve agent stores information about the event sieve object in a database and reads that information when the agent is started. See event filter and event forwarding discriminator.

event slot. In a Tivoli environment, a discrete area (a field) of an event record that contains a specific type of information about an event.

event specifier. In the Tivoli Enterprise Console, a rules-language program construct that is used to look for events in the event cache. For example, it can look for duplicate events, an event that matches a user-specified attribute, or an event that occurs within a certain time period. An event specifier is used in building rules, and it dictates how the Tivoli Enterprise Console will handle an event that it receives.

exception. An abnormal condition such as an I/O error encountered in processing a data set or a file.

exclusive set. In Remote Operations Service (ROPS), an option that indicates whether only the commands in the command list can be processed by ROPS or none of the commands in the command list can be processed by ROPS.

exclusive submap. In Tivoli NetView, a submap that is created by an application program wanting the exclusive right to control what happens in the application plane of the submap. Contrast with shared submap.

exec. (1) In the AIX operating system, to overlay the current process with another executable program. (2) See fork.

executable symbol. In Tivoli NetView, a symbol defined such that double-clicking on it causes an application program to perform an action on a set of target objects. Contrast with explodable symbol.

execution target. In a Tivoli environment, a managed node on which a job or other activity is performed. For example, if an application is being installed on a particular server, that server is the execution target for the installation activity.

explicit command. In Tivoli NetView for OS/390, a command that is used to request the display of information that the user would otherwise obtain by navigating through a hierarchy of panels.

explodable symbol. In Tivoli NetView, a symbol defined such that double-clicking on it or dragging and dropping it displays the child submap of the parent object that the symbol represents. Contrast with executable symbol.

export/import. In Tivoli Software Distribution, a feature that enables a Tivoli administrator to save (export) a file package definition as a text file, to edit the keywords and lists in the definition, and to retrieve (import) the definition from the text file to set the properties for the file package.

extended enterprise. The customers, suppliers, distributors, and other business partners with whom a company conducts e-business.

extranet. A private, virtual network that uses access control and security features to restrict the usage of one or more intranets attached to the Internet to selected subscribers (such as personnel from a sponsoring company and its business partners).

F

failover system. In Tivoli Manager for R/3, a computer that serves as a transparent backup to a primary computer. The primary computer and the failover

system share access to a common R/3 database, thereby enabling either machine to provide full database support.

fanout. In communication, the process of creating copies of a distribution to be delivered locally or to be sent through the network.

field. (1) An identifiable area in a window. Examples of fields are: an entry field, into which a user can type or place text, and a field of radio button choices, from which a user can select one choice. (2) The smallest identifiable part of a record. (3) In Tivoli NetView, the building block of which objects are composed. A field is characterized by a field name, a data type (integer, Boolean, character string, or enumerated value), and a set of flags that describe how the field is treated by Tivoli NetView. A field can contain data only when it is associated with an object.

field registration file (FRF). In Tivoli NetView, a file used to define fields for use in the object database.

file name substitution. In the AIX operating system, the process in which the shell substitutes an alphabetically sorted list of file names in the place of a pattern. The shell recognizes a pattern (as opposed to a file name) by the occurrence of a word (character string) with either of the following characteristics:

- The word contains any of these characters: *, ?, [, or {.
- The word begins with this character: ~.

file package. In Tivoli Software Distribution, a profile. The file package describes which files and directories to distribute and how to distribute them.

file package block. In Tivoli Software Distribution, a "snapshot" of a file package; that is, a static file containing (a) the file package definition (b) the file package attributes (c) the source files and directories, and (d) the configuration programs of a specific file package.

filter. (1) A device or program that separates data, signals, or material in accordance with specified criteria. (A) (2) In Tivoli NetView for OS/390, a function that limits the data recorded in the database or displayed at the terminal. See recording filter and viewing filter. (3) In the AIX operating system, a command that reads standard input data, modifies the data, and sends it to the display screen.

filter editor. In Tivoli NetView, a part of the graphical user interface (GUI) that enables the user to define, modify, and delete filtering rules for use by application programs.

firewall. In communication, a functional unit that protects and controls the connection of one network to other networks. The firewall (a) prevents unwanted or unauthorized communication traffic from entering the protected network and (b) allows only selected communication traffic to leave the protected network.

focal point (FP). (1) A system that provides centralized management services. (2) See management services focal point.

foreground process. (1) In the AIX operating system, a process that must run to completion before another command is issued to the shell. The foreground process is in the foreground process group, which is the group that receives the signals generated by a terminal. (2) Contrast with background process.

foreground task. The task with which the user is interacting. Contrast with background task.

foreign host. See remote host.

fork. In the UNIX operating system, to create and start a child process.

fpblock. See file package block.

FQDN. See fully qualified domain name.

fragmentation. An operating system's process of writing different parts of a file to discontiguous sectors on a computer storage medium when contiguous space that is large enough to contain the entire file is not available. When data is thus fragmented, the time that it takes to access the data may increase because the operating system must search different tracks for information that should be in one location. Contrast with defragmentation.

FRF. See field registration file.

full pathname. See absolute path.

fully qualified domain name (FQDN). In the Internet suite of protocols, the name of a host system that includes all of the subnames of the domain name. An example of a fully qualified domain name is `ralvm7.vnet.ibm.com`. See host name.

G

gadget. In the AIXwindows Toolkit, a windowless graphical object that looks like its equivalent like-named widget but does not support the translations, actions, or pop-up widget children supplied by that widget.

gateway. (1) A functional unit that interconnects two computer networks with different network architectures. A gateway connects networks or systems of different architectures. A bridge interconnects networks or systems with the same or similar architectures. (T) (2) A functional unit that connects two networks or subnetworks having different characteristics, such as different protocols or different policies concerning security or transmission priority. (3) The combination of machines and programs that provide address

translation, name translation, and system services control point (SSCP) rerouting between independent SNA networks to allow those networks to communicate. A gateway consists of one gateway NCP and at least one gateway VTAM. (4) In a Tivoli environment, software running on a managed node that provides all communication services between a group of endpoints and the rest of the Tivoli environment. This gateway includes the multiplexed distribution (MDist) function, enabling it to act as the fanout point for distributions to many endpoints. (5) See router.

gateway-capable host. A host node that has a defined NETID and SSCPNAME but does not perform gateway control functions, such as cross-network session initiation and termination.

gateway host. (1) A host node that contains a gateway system services control point (SSCP). See gateway-capable host. (2) In the AIX operating system, a host that connects independent networks. It has multiple interfaces, each with a different name and address.

gateway method. In a Tivoli environment, a method that runs on the gateway's proxy managed node on behalf of the endpoint. Results of the method are forwarded to the calling managed resource.

GCS. See graphic communication server.

GDDM®. See Graphical Data Display Manager.

GDDM interface for X Window System (GDDMXD). A graphical interface that formats and displays characters, graphics, and images on workstation display devices that support the X Window System.

GDDMXD. See GDDM interface for X Window System.

GDF. See global description file.

GDS. See graphic data server.

GEM. See Tivoli Global Enterprise Manager.

general topology manager (GTM). In Tivoli NetView, the component that accepts information about resources that are accessed through protocols other than the Internet Protocol (IP), stores this information in a database, and displays it to the user.

generic alert. In SNA management services (SNA/MS), alert information that is encoded using a method in which code points provide an index into short units of stored text. The use of generic alerts prevents the receiver from having to recognize and understand each unique problem for which an alert is sent. Contrast with non-generic alert.

generic collection. In a Tivoli environment, a collection that contains objects representing resources of any type.

GID. See group ID.

GIF. See graphical interchange format.

global description file (GDF). In the context of the Application Management Specification (AMS), an application description file that provides global information about an application such as the application name, the version identifier, and a free-form description of the application. Each version of a management-ready application is represented by one global description file.

GMFHS. See Graphic Monitor Facility host subsystem.

Graphical Data Display Manager (GDDM). In the NetView Performance Monitor (NPM), an IBM licensed program used in conjunction with the Presentation Graphics Feature (PGF) to generate online graphs in the NPM Graphic Subsystem.

graphical interchange format (GIF). A digital format that is used to compress and transfer graphical information over computer networks. For example, GIF is a common format for graphical information on the Internet.

graphical user interface (GUI). A type of computer interface consisting of a visual metaphor of a real-world scene, often of a desktop. Within that scene are icons, representing actual objects, that the user can access and manipulate with a pointing device. Contrast with command line interface.

graphic communication server (GCS). The part of the NetView Graphic Monitor Facility that manages LU 6.2 sessions used for data transport between (a) Tivoli NetView for OS/390 and the server workstation and (b) the server workstation and its client workstations.

graphic data server (GDS). The part of the NetView Graphic Monitor Facility that receives network management data from Tivoli NetView for OS/390, maintains this data (except for dynamically created view information), and correlates this data with views.

graphic monitor. The graphical user interface (GUI) component of the NetView Graphic Monitor Facility.

Graphic Monitor Facility host subsystem (GMFHS). In Tivoli NetView for OS/390, a component that manages updates to the configuration and status of resources displayed in NetView Graphic Monitor Facility (NGMF) views.

graphics context (GC, Gcontext). In the Enhanced X-Windows Toolkit, the storage area for various kinds of graphics output, such as foreground pixels, background pixels, line widths, and clipping regions. A graphics context can be used only with drawables that have the same root and the same depth as the graphics context.

graphics data file (GDF). A picture definition in a coded format that is used internally by the Graphical

Data Display Manager (GDDM) and, optionally, provides the user with a lower level program interface than the GDDM application programming interface (API).

group ID (GID). In the AIX operating system, a number that corresponds to a specific group name. The group ID can often be substituted in commands that take a group name as a value.

group profile. In Tivoli User Administration, a profile that a Tivoli administrator uses to define and modify information about a group of users.

GTM. See general topology manager.

GUI. See graphical user interface.

H

hardcoded. Pertaining to software instructions that are statically encoded and not intended to be altered.

hardcopy task (HCT). In Tivoli NetView for OS/390, the subtask that controls the passage of data between the NetView program and the hardcopy device.

hardware monitor. In Tivoli NetView for OS/390, the component that helps identify and solve problems related to physical network elements (as opposed to logical sessions, which are managed by the session monitor). Contrast with session monitor.

HCT. See hardcopy task.

heartbeat. In software products, a signal that one entity sends to another to convey that it is still active.

home submap. In Tivoli NetView, the first submap that appears when a map is opened. Each map has a home submap. When new maps are created, the home submap is the root submap.

hook. A location in a computer program where an instruction is inserted for invoking a particular function.

host. (1) A computer that is connected to a network (such as the Internet or an SNA network) and provides an access point to that network. Also, depending on the environment, the host may provide centralized control of the network. The host can be a client, a server, or both a client and a server simultaneously. (2) In a Tivoli environment, a computer that serves as a managed node for a profile distribution. (3) See host processor.

host name. In the Internet suite of protocols, the name given to a machine. Sometimes, “host name” is used to mean fully qualified domain name; other times, it is used to mean the most specific subname of a fully qualified domain name. For example, if `ralvm7.vnet.ibm.com` is the fully qualified domain name, either of the following may be considered the host name:

- `ralvm7.vnet.ibm.com`
- `ralvm7`

host namespace profile. In Tivoli Enterprise software, a profile that contains information about the list of hosts and their properties, such as host IP addresses and host aliases.

host processor. (1) A processor that controls all or part of a user application network. (T) (2) In a network, the processing unit in which the data communication access method resides.

host transit time. In the NetView Performance Monitor (NPM), the average time (in seconds) that all transactions spend in the host. It includes both VTAM and application time. It is also reported as an average for the transactions originating at the logical unit for which data collection is occurring.

HTML. See Hypertext Markup Language.

HTTP. See Hypertext Transfer Protocol.

hub. In a network, a point at which circuits are either connected or switched. For example, in a star network, the hub is the central node; in a star/ring network, it is the location of wiring concentrators.

Hypertext Markup Language (HTML). A markup language that is specified by an SGML document type definition (DTD) and is understood by all Web servers.

Hypertext Transfer Protocol (HTTP). In the Internet suite of protocols, the protocol that is used to transfer and display hypertext documents.

I

IAB. See Internet Architecture Board.

ICMP. See Internet Control Message Protocol.

IDL. See Interface Definition Language.

IETF. See Internet Engineering Task Force.

immediate command. In Tivoli NetView for OS/390, a command (such as GO, RESET, or LOGOFF) that begins processing as soon as the operator enters it, possibly preempting other ongoing processing. All other commands are called ″regular commands″ and are processed by a ″regular command processor.″ Regular commands can run concurrently with other regular commands and can be interrupted by immediate commands. Most commands and all command lists are regular commands.

IMS™. See Internet Management Specification.

indicator. In Tivoli Distributed Monitoring, an icon on the Tivoli desktop that graphically displays the status of a monitor that has been associated with it. The icon

resembles a thermometer, which the Tivoli administrator can read to determine the status of the monitor.

indicator collection. In a Tivoli environment, a single location from which a Tivoli administrator can determine the status of monitors in different profiles, as well as clear and reset alarmed states.

instance. In object-oriented programming, an object created by instantiating a class.

instantiate. In object-oriented programming, to represent a class abstraction with a concrete instance of the class.

instrument. In application or system software, to use monitoring functions to provide performance and other information to a management system.

instrumentation. In application or system software, either (a) monitoring functions that provide performance and other information to a management system or (b) the use of monitoring functions to provide performance and other information to a management system.

intelligent agent. Software that monitors conditions or actions on a network node and contains logic enabling it to respond to these conditions or actions.

interactive chart utility (ICU). A utility provided by the Graphical Data Display Manager (GDDM) to allow basic graphic handling capability and a menu-driven generation of different forms of graphs. ICU is a part of the presentation graphics feature.

Interface Definition Language (IDL). In CORBA, a declarative language that is used to describe object interfaces, without regard to object implementation.

Internet Architecture Board (IAB). The technical body that oversees (at a high level) the work of the Internet Engineering Task Force (IETF). The IAB approves the membership of the IETF.

Internet Control Message Protocol (ICMP). The protocol used to handle errors and control messages in the Internet Protocol (IP) layer. Reports of problems and incorrect datagram destinations are returned to the original datagram source.

Internet Engineering Task Force (IETF). The task force of the Internet Architecture Board (IAB) that is responsible for solving the short-term engineering needs of the Internet. The IETF consists of numerous working groups, each focused on a particular problem. Internet standards are typically developed or reviewed by individual working groups before they can become standards.

Internet Management Specification (IMS). A draft specification for an open standard for managing Internet resources and services.

internet object. In Tivoli NetView, a node or a network that can be accessed by the Internet Protocol (IP).

Internet Protocol (IP). In the Internet suite of protocols, a connectionless protocol that routes data through a network or interconnected networks. IP acts as an intermediary between the higher protocol layers and the physical network. However, this protocol does not provide error recovery and flow control and does not guarantee the reliability of the physical network.

Internet service provider (ISP). An organization that provides access to the Internet.

Internetwork Packet Exchange (IPX). The network protocol used to connect Novell's servers, or any workstation or router that implements IPX, with other workstations. Although similar to the Internet Protocol (IP), IPX uses different packet formats and terminology.

interprocess communication (IPC). The process by which programs communicate data to each other and synchronize their activities. Semaphores, signals, and internal message queues are common methods of interprocess communication.

intranet. A private network that integrates Internet standards and applications (such as Web browsers) with an organization's existing computer networking infrastructure.

IP. See Internet Protocol.

IPC. See interprocess communication.

IPX. See Internetwork Packet Exchange.

ISP. See Internet service provider.

IT. Information technology.

J

Java. An object-oriented programming language for portable interpretive code that supports interaction among remote objects. Java was developed and specified by Sun Microsystems, Incorporated.

JavaBeans. A platform-independent, software component technology for building reusable Java components called "beans." Once built, these beans can be made available for use by other software engineers or can be used in Java applications. Also, using JavaBeans, software engineers can manipulate and assemble beans in a graphical drag-and-drop development environment.

Java Database Connectivity (JDBC). An application programming interface (API) that has the same characteristics as Open Database Connectivity (ODBC) but is specifically designed for use by Java database applications. Also, for databases that do not have a JDBC driver, JDBC includes a JDBC to ODBC bridge,

which is a mechanism for converting JDBC to ODBC; it presents the JDBC API to Java database applications and converts this to ODBC. JDBC was developed by Sun Microsystems, Inc. and various partners and vendors.

Java Management Application Programming Interface (JMAPI). A specification proposed by Sun Microsystems that defines a core set of application programming interfaces for developing tightly integrated system, network, and service management applications. The application programming interfaces could be used in diverse computing environments that encompass many operating systems, architectures, and network protocols.

JDBC. See Java Database Connectivity.

JMAPI. See Java Management Application Programming Interface.

job. (1) A unit of work defined by a user that is to be accomplished by a computer. Loosely, the term job is sometimes used to refer to a representation of a job. This representation may include a set of computer programs, files, and control statements to the operating system. (I) (A) (2) A Printing Systems Manager (PSM) object that represents a request to print one or more documents in a single printing session. (3) In a Tivoli environment, a resource consisting of a task and its preconfigured execution parameters. Among other things, the execution parameters specify the set of hosts on which the job is to execute.

JPEG. A standard format for storing compressed true-color images. ″JPEG″ represents ″Joint Photographic Experts Group,″ which is the name of the committee that developed this standard format.

K

Kerberos. The security system of the Massachusetts Institute of Technology's (MIT's) Project Athena. It uses symmetric key cryptography to provide security services to users in a network.

Kerberos master machine. In Kerberos, the host machine on which the Kerberos database resides.

Kerberos master password. In Kerberos, the password required to change or access the Kerberos database.

Kerberos principal. In Kerberos, a service or user that is known to the Kerberos system. See principal name.

Kerberos realm. In Kerberos, a set of managed nodes that share the same Kerberos database.

key. In computer security, a sequence of symbols that is used with a cryptographic algorithm for encrypting or decrypting data. See private key and public key.

keyword. (1) In programming languages, a lexical unit that, in certain contexts, characterizes some language construct; for example, in some contexts, IF characterizes an if-statement. A keyword normally has the form of an identifier. (I) (2) One of the predefined words of an artificial language. (A) (3) A name or symbol that identifies a parameter. (4) The part of a command operand that consists of a specific character string (such as `DSNAME=`). (5) See keyword operand.

keyword operand. (1) An operand that consists of a keyword followed by one or more values (such as `DSNAME=HELLO`). (2) Contrast with positional operand. (3) See definition statement.

keyword parameter. A parameter that consists of a keyword followed by one or more values.

L

LAN. See local area network.

LAN Network Manager (LNM). An IBM licensed program that enables a user to manage and monitor LAN resources from a central workstation.

LCCM. See link connection component manager.

LCSM. See link connection subsystem manager.

link connection component manager (LCCM). The transaction program that manages the configuration of the link connection.

link connection subsystem manager (LCSM). The transaction program that manages the sequence of link connection components that belong to a link connection.

Link Problem Determination Aid (LPDA®). A series of procedures that are used to test the status of and to control DCEs, the communication line, and the remote device interface. These procedures, or a subset of them, are implemented by host programs (such as Tivoli NetView for OS/390 and VTAM), communication controller programs (such as NCP), and IBM LPDA DCEs. See LPDA-1 and LPDA-2.

LNM. See LAN Network Manager.

local area network (LAN). (1) A computer network located on a user's premises within a limited geographical area. Communication within a local area network is not subject to external regulations; however, communication across the LAN boundary may be subject to some form of regulation. (T) (2) A network in which a set of devices are connected to one another for communication and that can be connected to a larger network.

local distribution. In a Tivoli environment, a distribution to target machines in the same Tivoli Management Region as the source machine.

local overrides. In a Tivoli environment, a feature of all profile-based Tivoli applications—except for Tivoli Software Distribution—that allows changes made at the endpoint profile to override those in a distributed profile.

local registration file (LRF). In Tivoli NetView, a file that provides information about an agent or daemon, such as the name, the location of the executable code, the names of processes dependent on the agent or daemon, and details about the objects that an agent manages.

local topology database. A database in an APPN or LEN node containing an entry for each transmission group (TG) having at least one end node for an endpoint. In an end node, the database has one entry for each TG connecting to the node. In a network node, the database has an entry for each TG connecting the network node to an end node. Each entry describes the current characteristics of the TG that it represents. A network node has both a local and a network topology database while an end node has only a local topology database.

lock. The means by which integrity of data is ensured by preventing more than one user from accessing or changing the same data or object at the same time.

logged-on operator. (1) In Tivoli NetView for OS/390, an operator station task that requires a terminal and a logged-on user. (2) Contrast with autotask.

LPDA. See Link Problem Determination Aid.

LPDA-1. The first version of the Link Problem Determination Aid (LPDA) command set. LPDA-1 is not compatible with LPDA-2.

LPDA-2. The second version of the Link Problem Determination Aid (LPDA) command set. LPDA-2 provides all of the functions of LPDA-1; it also supports commands such as the following:

- DCE configuration
- Dial
- Set transmit speed
- Commands to operate a contact that can control external devices.

LRF. See local registration file.

LUC session. Communication, using LU type 0 protocols, between the LUC tasks of two Tivoli NetView for OS/390 programs. This communication is similar to an LU 6.2 conversation.

LUC task. A Tivoli NetView for OS/390 task, denoted by the NetView domain ID concatenated with the literal "LUC" (for example, CNM01LUC), that serves as the endpoint of an LUC session.

LU group. (1) In the NetView Performance Monitor (NPM), a file containing a list of related or unrelated logical units. The LU group is used to help simplify data collection and analysis. (2) In Tivoli NetView for OS/390, a grouping of logical units according to some affinity, such as their link to the same VTAM generic resource or VTAM USERVAR.

LU 6.2 verb. A syntactical unit in the LU 6.2 application programming interface representing an operation.

M

macroinstruction. (1) An instruction in a source language that is to be replaced by a defined sequence of instructions in the same source language and that may also specify values for parameters in the replaced instructions. (T) (2) In assembler programming, an assembler language statement that causes the assembler to process a predefined set of statements called a macro definition. The statements normally produced from the macro definition replace the macroinstruction in the program.

managed node. (1) In Internet communications, a workstation, server, or router that contains a network management agent. In the Internet Protocol (IP), the managed node usually contains a Simple Network Management Protocol (SNMP) agent. (2) In a Tivoli environment, any managed resource on which the Tivoli Management Framework is installed.

managed object. (1) A component of a system that can be managed by a management application. (2) The systems management view of a resource that can be managed through the use of systems management protocols.

managed resource. In a Tivoli environment, any hardware or software entity (machine, service, system, or facility) that is represented by a database object and an icon on the Tivoli desktop. Managed resources must be a supported resource type in a policy region and are subject to a set of rules. Managed resources include, but are not limited to, managed nodes, task libraries, monitors, profiles, and bulletin boards.

management by subscription. In a Tivoli environment, the concept of managing network resources by creating sets of profiles and distributing the profiles (through profile managers) to physical entities (Tivoli resources), called subscribers.

Management Information Base (MIB). (1) A collection of objects that can be accessed by means of a network management protocol. (2) A definition for management information that specifies the information available from a host or gateway and the operations allowed. (3) In

OSI, the conceptual repository of management information within an open system. (4) See MIB module.

Management Information Format (MIF). The Desktop Management Interface (DMI) specification that defines the syntax for describing management information about the hardware and software components that can be installed on a computer system.

management module. In a Tivoli environment, a file that contains the management information and instrumentation for enabling a particular application or business system to be managed by Tivoli management software. This file may be in the form of a Tivoli install image or an application management package. Types of management modules include base modules, Tivoli GEM modules, and Tivoli Plus modules. See Tivoli Module Builder and Tivoli Module Designer.

management region. In Tivoli NetView, the set of managed objects on a particular map that defines the extent of the network that is being actively managed. The management region may vary across maps.

management services (MS). (1) One of the types of network services in control points (CPs) and physical units (PUs). Management services are the services provided to assist in the management of SNA networks, such as problem management, performance and accounting management, configuration management, and change management. (2) Services that assist in the management of systems and networks in areas such as problem management, performance management, business management, operations management, configuration management, and change management.

management services focal point (MSFP). For any given management services discipline (for example, problem determination or response time monitoring), the control point that is responsible for that type of network management data for a sphere of control. This responsibility may include collecting, storing, or displaying the data, or all of these. (For example, a problem determination focal point is a control point that collects, and that may store or display, problem determination data.)

manager. (1) In systems management, a user that, for a particular interaction, has assumed a manager role. (2) An entity that monitors or controls one or more managed objects by (a) receiving notifications regarding the objects and (b) requesting management operations to modify or query the objects. (3) A system that assumes a manager role.

manager role. In systems management, a role assumed by a user where the user is capable of issuing management operations and of receiving notifications.

man page. In UNIX systems, one page of online documentation. ″Man page″ is an abbreviation for ″manual page.″ Each UNIX command, utility, and library function has an associated man page that can be viewed by entering this command: `man` *`command name`*.

map. In Tivoli NetView, a database represented by a set of related submaps that provide a graphical and hierarchical presentation of a network and its systems.

mapper. In Tivoli NetView for OS/390, a function that records errors from resources attached to a communication controller or from certain channel-attached devices.

marshall. To copy data into a remote procedure call (RPC) packet. Stubs perform marshalling. Contrast with unmarshall.

MCSL. See Monitoring Collection Specification Language.

MDist. Multiplexed distribution. In a Tivoli environment, a service that enables efficient distribution of large amounts of data across complex networks.

menu bar. (1) The area near the top of a window, below the title bar and above the rest of the window, that contains choices that provide access to other menus. (2) In the AIX operating system, a rectangular area at the top of the client area of a window that contains the titles of the standard pull-down menus for that application.

message style. In Tivoli Distributed Monitoring, the amount and format of information presented by certain monitors.

method. (1) In object-oriented design or programming, the software that implements the behavior specified by an operation. (2) In Tivoli NetView for OS/390, a program that runs in the Resource Object Data Manager (RODM) address space and communicates with RODM using an application programming interface (API). Methods are usually small programs that perform specific tasks on data in the data cache.

MIB. See Management Information Base.

MIB application program. A systems management application program used to monitor network devices.

MIB module. In the Simple Network Management Protocol (SNMP), a collection of objects relating to a common management area. See MIB variable.

MIB object. See MIB variable.

MIB tree. In the Simple Network Management Protocol (SNMP), the structure of the Management Information Base (MIB).

MIB variable. In the Simple Network Management Protocol (SNMP), a specific instance of data defined in a MIB module.

MIB view. In the Simple Network Management Protocol (SNMP), the collection of managed objects, known to the agent, that is visible to a particular community.

MIB walking. In the Simple Network Management Protocol (SNMP), a technique of looking for Management Information Base (MIB) tree information when it is presented in a hierarchical format.

Mid-Level Manager (MLM). In Tivoli NetView, the component that performs certain systems and network management tasks (for example, polling, status monitoring, and node discovering) for a defined set of Simple Network Management Protocol (SNMP) devices in the network, thereby offloading these tasks from Tivoli NetView.

MIF. See Management Information Format.

MIPS. A measure of computer processing performance that is equal to one million instructions per second.

MLM. See Mid-Level Manager.

MLM Configuration Application. A Tivoli NetView feature that is used to configure the Mid-Level Manager (MLM).

MNPS. See multinode persistent session.

module. See management module.

monitor. (1) A device that observes and records selected activities within a data processing system for analysis. Possible uses are to indicate significant departure from the norm, or to determine levels of utilization of particular functional units. (T) (2) Software or hardware that observes, supervises, controls, or verifies operations of a system. (A) (3) Software that monitors specific applications or the systems on which the applications rely. Monitors typically monitor information such as available disk space or application errors and compare the information to defined thresholds. When thresholds are exceeded, either system or network administrators can be notified, or an automated response can be performed. (4) In the NetView Graphic Monitor Facility, to open a view that can receive status changes from Tivoli NetView for OS/390. Problem determination and correction can be performed directly from the view. Contrast with browse.

monitoring collection. In Tivoli Distributed Monitoring, a collection of predefined monitors. Several monitoring collections are packaged with Tivoli Distributed Monitoring, but Tivoli administrators can use custom-developed and third-party monitoring collections as well. See custom monitor.

Monitoring Collection Specification Language (MCSL). A proprietary programming language that is owned by Tivoli Systems Inc. and is used to define monitoring collections for Tivoli Distributed Monitoring.

MPM. See MultiPlatform Manager.

MS. See management services.

MSFP. See management services focal point.

multinode persistent session (MNPS). An LU-LU session that is retained after the failure of VTAM, the operating system, or the hardware.

MultiPlatform Manager (MPM). An application programming interface (API) that was developed by a group of leading technology vendors, including Tivoli Systems Inc., and that enables disparate management systems to interoperate with each other. Tivoli LAN Access and Tivoli IT Director support this API, which means that Tivoli Enterprise software or Tivoli IT Director can provide IT managers with unifying, centralized control over disconnected management resources.

Multiple Virtual Storage/Operator Communication Control Facility (MVS/OCCF). A facility that intercepts messages from the MVS supervisor. Tivoli NetView for OS/390 and MVS/OCCF help a network operator control multiple MVS systems from a central site.

multiplexed distribution (MDist). See MDist.

MultiSystem Manager. In Tivoli NetView for OS/390, the component that manages non-SNA resources, such as those in IP networks, NetWare networks, LAN Network Manager networks, and LAN NetView Management Utilities networks.

multitiered application. An application that is deployed on more than one physical machine. A client/server application is a common multitiered application in which there are two tiers: the client tier (for example, the presentation and the graphical user interface) and the server tier (for example, the service and the database).

MVS/OCCF. See Multiple Virtual Storage/Operator Communication Control Facility.

MVS system symbol. In a sysplex where a customer runs a copy of a given program (such as CICS or Tivoli NetView for OS/390) on more than one MVS image, a symbol that the customer can use to write generic JCL for use by each instance of the given program. An MVS system symbol behaves like a program variable that the sysplex resolves at execution time with the value that is appropriate to the MVS image on which the program instance is running.

N

name registry. In a Tivoli environment, a name service consisting of a two-dimensional table that maps

resource names to resource identifiers and corresponding information within a Tivoli Management Region.

name translation. In SNA network interconnection, the conversion of logical unit names, logon mode table names, and class-of-service names used in one network to equivalent names for use in another network. This function can be provided through Tivoli NetView for OS/390 and invoked by a gateway system services control point (SSCP) when necessary. See alias name.

NAT. See network address translation.

navigate. In the NetView Graphic Monitor Facility, to move between levels in the view hierarchy.

navigation tree. In Tivoli NetView, a component of the graphical user interface (GUI) that displays a hierarchy of open submaps illustrating the parent-child relationship. The navigation tree enables the network operator to determine which submaps are currently open and to close, restore, or raise the windows that contain submaps.

NCCF. In Tivoli NetView for OS/390, a command that starts the NetView command facility. Also, the use of the abbreviation “NCCF” indicates that various panels and functions are part of the command facility.

nested file package. In Tivoli Software Distribution, a file package that is added as an entry to another file package.

NetBIOS. (1) Network Basic Input/Output System. A standard interface to networks, IBM personal computers (PCs), and compatible PCs, that is used on LANs to provide message, print-server, and file-server functions. Application programs that use NetBIOS do not need to handle the details of LAN data link control (DLC) protocols. (2) See Basic Input/Output System.

NetView. See Tivoli NetView and Tivoli NetView for OS/390.

NetView AutoBridge. In Tivoli Service Desk for OS/390, an application interface to Tivoli NetView for OS/390 that works with the NetView Bridge Adapter to update the Tivoli Service Desk for OS/390 database and to automate network monitoring. The NetView AutoBridge receives data from NetView alerts, messages, and other applications and uses this data to build and perform Tivoli Service Desk for OS/390 transactions.

NetView Bridge. In Tivoli NetView for OS/390, a set of application programming interfaces (APIs) that enable Tivoli NetView for OS/390 to interact with various types of databases in the OS/390 environment.

NetView Bridge Adapter. In Tivoli Service Desk for OS/390, a feature that provides a connection between the NetView Bridge and the Tivoli Service Desk for OS/390 database. The NetView Bridge Adapter enables the Tivoli Service Desk for OS/390 to act as a NetView database server and works with the NetView AutoBridge or other NetView applications to access problem records logged in the Tivoli Service Desk for OS/390 database.

NetView command authorization table. In Tivoli NetView for OS/390, a set of entries that define an operator’s authorization for accessing commands and (depending on the level of granularity that an enterprise chooses) command keywords and keyword values.

NetView command list language. In Tivoli NetView for OS/390, an interpretive language that is unique to the NetView program and that is used to write NetView command lists in environments where REXX is not supported.

NetView Graphic Monitor Facility (NGMF). In Tivoli NetView for OS/390, a function that provides the network operator with a graphical topological representation of a network and allows the operator to manage the network interactively.

NetView help desk. In Tivoli NetView for OS/390, an online information facility that guides the help desk operator through problem management procedures.

NetView Installation and Administration Facility/2 (NIAF/2). An OS/2-based tool that allows new users of Tivoli NetView for OS/390 or users migrating from a prior release to install, administer, and maintain Tivoli NetView for OS/390. NIAF/2 replaces the Interactive System Productivity Facility-based (ISPF-based) NetView Installation Facility.

NetView management console. See topology console.

NetView management console server. See topology server.

NetView-NetView task (NNT). In Tivoli NetView for OS/390, the task under which a cross-domain NetView operator session runs. See operator station task.

NetView Performance Monitor (NPM). An IBM licensed program that collects, monitors, analyzes, and displays data relevant to the performance of a VTAM telecommunication network. It runs as an online VTAM application program.

NetWare managed site. In a Tivoli environment, a resource that represents (a) a Novell NetWare server on which the Tivoli NetWare repeater (TNWR) is installed and (b) one or more clients. A NetWare managed site enables profiles to be distributed through the NetWare server to one or more specified client PCs using either TCP/IP or IPX.

network address translation (NAT). In a firewall, the conversion of secure IP addresses to external

registered addresses. This enables communication with external networks but masks the IP addresses that are used inside the firewall.

network class. In Tivoli NetView, an object class used for symbols that represent compound objects that may contain objects such as hosts and network devices. Contrast with connector class.

network computing. The use of a scalable distributed computing infrastructure that encompasses the key elements of today's networking technologies, such as systems and network management; the Internet and intranets; clients and servers; application programs; databases; transaction processing; and various operating systems and communication protocols.

Network File System (NFS). A protocol developed by Sun Microsystems, Incorporated, that allows any host in a network to mount another host's file directories. Once mounted, the file directory appears to reside on the local host.

network gateway accounting (NGA). The NetView Performance Monitor (NPM) subsystem that receives traffic information from the gateway NCP for sessions that flow throughout a network.

Network Information Center (NIC). In Internet communications, local, regional, and national groups throughout the world who provide assistance, documentation, training, and other services to users.

Network Information Services (NIS). A set of UNIX network services (for example, a distributed service for retrieving information about the users, groups, network addresses, and gateways in a network) that resolve naming and addressing differences among computers in a network.

network log. A file that contains (a) messages, commands, and command procedures that have been processed by Tivoli NetView for OS/390 and (b) output resulting from commands, command procedures, and other activity occurring within Tivoli NetView for OS/390.

network management gateway (NMG). A gateway between Tivoli NetView for OS/390, which is the SNA network management system, and the network management function of one or more non-SNA networks.

network management vector transport (NMVT). A management services request/response unit (RU) that flows over an active session between physical unit management services and control point management services (SSCP-PU session).

Network News Transfer Protocol (NNTP). In the Internet suite of protocols, a protocol for the distribution, inquiry, retrieval, and posting of news articles that are stored in a central database.

network session accounting (NSA). The NetView Performance Monitor (NPM) subsystem that receives session accounting information from the NCP for sessions that flow throughout a network.

network topology database. The representation of the current connectivity between the network nodes within an APPN network. It includes (a) entries for all network nodes and the transmission groups interconnecting them and (b) entries for all virtual routing nodes to which network nodes are attached.

NFS. See Network File System.

NFS client. A program or system that mounts remote file directories from another host called a Network File System (NFS) server.

NFS server. A program or system that allows authorized remote hosts called Network File System (NFS) clients to mount and access its local file directories.

NGA. See network gateway accounting.

NGMF. See NetView Graphic Monitor Facility.

NIAF/2. See NetView Installation and Administration Facility/2.

NIC. See Network Information Center.

NIS. See Network Information Services.

NLDM. In Tivoli NetView for OS/390, a command that starts the session monitor. Also, the use of the abbreviation "NLDM" indicates that various panels and functions are part of the session monitor.

NMG. See network management gateway.

NMVT. See network management vector transport.

NNT. See NetView-NetView task.

NNTP. See Network News Transfer Protocol.

non-generic alert. In SNA management services (SNA/MS), alert information that is encoded such that it conveys to the receiver the set of screens that should be displayed for the network operator when the alert is received. The use of non-generic alerts requires that the receiver recognize and understand each unique problem for which an alert is sent. Contrast with generic alert.

NOS. Network operating system.

notice. In a Tivoli environment, a message generated by a systems management operation that contains information about an event or the status of an application. Notices are stored in notice groups. See bulletin board.

notice group. In a Tivoli environment, an application- or operation-specific container that stores and displays notices pertaining to specific Tivoli functions. The Tivoli bulletin board is comprised of notice groups. A Tivoli administrator can subscribe to one or more notice groups; the administrator's bulletin board contains only the notices that reside in a notice group to which the administrator is subscribed.

notification. (1) An unscheduled, spontaneously generated report of an event that has occurred. (2) In systems management, information emitted by a managed object relating to an event that has occurred within the managed object, such as a threshold violation or a change in configuration status.

NPALU. In the NetView Performance Monitor (NPM), the virtual logical unit generated in an NCP with which the network subsystem communicates.

NPDA. In Tivoli NetView for OS/390, a command that starts the hardware monitor. Also, the use of the abbreviation "NPDA" indicates that various panels and functions are part of the hardware monitor.

NPM. See NetView Performance Monitor.

NSA. See network session accounting.

NT repeater. In a Tivoli environment, the first Windows NT machine on which the Tivoli Remote Execution Service is installed. Using fanout, the NT repeater distributes the Tivoli Remote Execution Service to all other NT clients during the client installation process.

null resource. In the NetView Graphic Monitor Facility, an object that is used only as an aid in formatting and drawing a view. A null resource always shows the status "unknown."

O

object. (1) In object-oriented design or programming, a concrete realization of a class that consists of data and the operations associated with that data. (2) An item that a user can manipulate as a single unit to perform a task. An object can appear as text, an icon, or both. (3) In Tivoli NetView for OS/390, the part of a NetView command that follows the verb. The object describes where the action of the verb is to occur.

object dispatcher. See object request broker.

object identifier (OID). An administratively assigned data value of the type defined in abstract syntax notation 1 (ASN.1).

Object Management Group (OMG). A non-profit consortium whose purpose is to promote object-oriented technology and the standardization of that technology. The Object Management Group was formed to help reduce the complexity, lower the costs, and hasten the introduction of new software applications.

object path. In a Tivoli environment, an absolute or relative path to a Tivoli object, similar to paths in file systems.

object reference. In a Tivoli environment, the object identifier (OID) given to an object during its creation.

object registration service (ORS). In Tivoli NetView, a component that creates and maintains a global directory of object managers, their locations, and their protocols. The postmaster daemon uses this directory to route messages and provide location transparency for managers and agents.

object request broker (ORB). In object-oriented programming, software that serves as an intermediary by transparently enabling objects to exchange requests and responses. See Common Object Request Broker Architecture.

ODBC. See Open Database Connectivity.

OID. See object identifier.

OMG. See Object Management Group.

Open Database Connectivity (ODBC). A standard application programming interface (API) for accessing data in both relational and nonrelational database management systems. Using this API, database applications can access data stored in database management systems on a variety of computers even if each database management system uses a different data storage format and programming interface. ODBC is based on the call level interface (CLI) specification of the X/Open SQL Access Group and was developed by Digital Equipment Corporation (DEC), Lotus®, Microsoft®, and Sybase. Contrast with Java Database Connectivity.

operation. In object-oriented design or programming, a service that can be requested at the boundary of an object. Operations include modifying an object or disclosing information about an object.

operations and administration. The Tivoli management discipline that addresses the automation of activities that ensure the operational integrity and reliability of a network computing system. See availability management, deployment management, and security management.

operator. A person or a program that manages activities that are controlled by a specific computer program.

operator profile. In Tivoli NetView for OS/390, a specification of the resources and activities over which a network operator has control. The profile is stored in a file that is activated when the operator logs on.

operator station task (OST). In Tivoli NetView for OS/390, the task that establishes and maintains the online session with the network operator. There is one operator station task for each network operator who logs on to Tivoli NetView for OS/390. See NetView-NetView task.

ORB. See object request broker.

ORS. See object registration service.

oserv. The name of the object request broker used by the Tivoli environment. `0serv` runs on the TMR server and each TMR client.

OST. See operator station task.

P

package definition file (PDF). In Tivoli IT Director, an ASCII text file that contains predefined workstation, sharing, and inventory property settings for a file package.

packet. In data communication, a sequence of binary digits, including data and control signals, that is transmitted and switched as a composite whole. The data, control signals, and, possibly, error control information are arranged in a specific format. (I)

parameter. (1) A variable that is given a constant value for a specified application and that may denote the application. (I) (A) (2) In Common User Access (CUA®) architecture, a variable used in conjunction with a command to affect its result. (3) An item in a menu for which the user specifies a value or for which the system provides a value when the menu is interpreted. (4) Data passed to a program or procedure by a user or another program, namely as an operand in a language statement, as an item in a menu, or as a shared data structure. (5) In Tivoli NetView for OS/390, a part of a command's object. (6) See keyword and keyword parameter.

parent process. In the UNIX operating system, a process that creates other processes. See child process and fork.

parent resource. In the NetView Graphic Monitor Facility, a resource that has one or more child resources below it in a hierarchy.

PassTicket. In RACF® secured sign-on, a dynamically generated, random, one-time-use, password substitute that a workstation or other client can use to sign on to the host rather than sending a RACF password across the network.

PassTicket application key. In RACF secured sign-on, an encryption key that is used in the creation and evaluation of a PassTicket. The PassTicket application key is sometimes referred to as the "secured sign-on application key."

patch. A code change that is sent to the owners of a software product license after the release of a product. The licensees can then apply this code change to correct a reported problem.

path. (1) A list of one or more directory names and an object name (such as the name of a file) that are separated by an operating system-specific character, such as the slash (/) in UNIX operating systems, the backslash (\) in Windows® operating systems, and the semicolon (;) in OS/2 operating systems. The directory names detail the path to follow, in left-to-right order, to locate the object within the file system. This concept of path is also known as the "pathname." (2) A list of directory names, usually separated by a colon (:), that are to be searched (in left-to-right order) to locate an object. This concept of path is also known as the "search path." (3) See absolute path, directory, relative path, root directory, and working directory.

pathname. See path.

path test. A test provided by Tivoli NetView for OS/390 that enables a network operator to determine whether a path is available between two LUs that are currently in session.

pattern-matching character. A special character such as an asterisk (*) or a question mark (?) that can be used to represent one or more characters. Any character or set of characters can replace a pattern-matching character.

PC agent. In a Tivoli environment, software installed on a client PC that enables Tivoli operations to execute on the PC. See PC managed node.

PC managed node. In a Tivoli environment, an object that represents a client PC. The Tivoli Management Framework can communicate with the client PC only if the PC agent is installed on the PC. Client PCs are most often referred to as PC managed nodes.

PDF. (1) See package definition file. (2) See Portable Document Format.

performance class. In Tivoli NetView for OS/390, a description of an objective or commitment of performance. It consists of a performance class name, boundary definitions, response time definition, response time ranges, and response time percentage objectives. Sessions may be assigned performance classes.

persistent LU-LU session. See persistent session.

persistent session. (1) In Tivoli NetView for OS/390, a network management session that remains active even though there is no activity on the session for a specified period of time. (2) An LU-LU session that

VTAM retains after the failure of a VTAM application program. Following the application program's recovery, the application program restores or terminates the session. This session is sometimes referred to as a "single-node persistent session." See multinode persistent session.

pipeline. (1) A serial arrangement of processors or a serial arrangement of registers within a processor. Each processor or register performs part of a task and passes results to the next processor; several parts of different tasks can be performed at the same time. (2) To perform processes in series. (3) To start execution of an instruction sequence before the previous instruction sequence is completed to increase processing speed. (4) In Tivoli NetView for OS/390, a message processing procedure that consists of one or more programs known as stages.

pixel map. (1) A three-dimensional array of bits. A pixel map can be thought of as a two-dimensional array of pixels, with each pixel being a value from zero to 2 to the power N -1, where N is the depth of the pixel map. (2) In the X Window System, a data type to which icons, originally created as bitmaps, are converted.

pixmap. See pixel map.

platform. An ambiguous term that may refer to the hardware, the operating system, or a combination of the hardware and the operating system on which software programs run.

plex. A Printing Systems Manager (PSM) attribute used for defining the capability of a printer to support different placements of output images on a medium. For example, the plex attribute could specify whether the printer is to support simplex or tumble mode.

Plus module. See Tivoli Plus module.

policy. In a Tivoli environment, a set of rules that are applied to managed resources. A specific rule in a policy is referred to as a "policy method."

policy region. In a Tivoli environment, a group of managed resources that share one or more common policies. Tivoli administrators use policy regions to model the management and organizational structure of a network computing environment. The administrators can group similar resources, define access to and control the resources, and associate rules for governing the resources. The policy region contains resource types and the list of resources to be managed. A policy region is represented on the Tivoli desktop by an icon that resembles a capitol building (dome icon). When a Tivoli Management Region (TMR) is created, a policy region with the same name is also created. In this case, the TMR has only one policy region. However, in most cases, a Tivoli administrator creates other policy regions and subregions to represent the organization of the TMR. A TMR addresses the physical connectivity of resources whereas a policy region addresses the logical organization of resources.

policy subregion. In a Tivoli environment, a policy region created or residing in another policy region. When a policy subregion is created, it initially uses the resource and policy properties of the parent policy region. The Tivoli administrator can later change or customize these properties to reflect the specific needs and differences of the subregion.

polling. (1) On a multipoint connection or a point-to-point connection, the process whereby data stations are invited, one at a time, to transmit. (I) (2) Interrogation of devices for such purposes as to avoid contention, to determine operational status, or to determine readiness to send or receive data. (A) (3) In network management, the process by which a manager interrogates one or more managed nodes at regular intervals.

populate. In a Tivoli environment, to fill a profile with information that is to be distributed to the subscribing managed resources.

port. To modify a computer program to enable it to run on a different platform.

Portable Document Format (PDF). A standard specified by Adobe Systems, Incorporated, for the electronic distribution of documents. PDF files are compact; can be distributed globally via e-mail, the Web, intranets, or CD-ROM; and can be viewed with the Acrobat Reader, which is software from Adobe Systems that can be downloaded at no cost from the Adobe Systems home page.

portmapper. A program that maps client programs to the port numbers of server programs. Portmapper is used with remote procedure call (RPC) programs.

positional operand. (1) An operand in a language statement that has a fixed position. (2) Contrast with keyword operand. (3) See definition statement.

postmaster. In Tivoli NetView, a process (daemon) that directs network management information between multiple application programs and agents running concurrently. The postmaster determines the route by using specified addresses or a routing table that is configured in the object registration service.

PPI. See program-to-program interface.

presentation graphics feature (PGF). In the NetView Performance Monitor (NPM), a feature used in conjunction with the Graphical Data Display Manager (GDDM) to generate online graphs in the NPM graphic subsystem.

presentation services command processor (PSCP). In Tivoli NetView, a facility that processes requests from a user terminal and formats displays to be presented at the user terminal.

primary database. In Tivoli NetView for OS/390, the main database provided to the NetView user for recording error data. See secondary database.

primary POI task (PPT). In Tivoli NetView for OS/390, the subtask that processes all unsolicited messages that are received from the VTAM program operator interface (POI) and delivers them to the controlling operator or to the command processor. The PPT also processes (a) the initial command that is specified to execute when NetView is initialized and (b) timer request commands that are scheduled to execute under the PPT.

primary window. In OSF/Motif, the top-level window in an application program that can be minimized or represented by an icon. See submap window.

principal name. (1) In Kerberos, the name by which the Kerberos principal is identified. The principal name consists of three parts: a service or user name, an instance name, and a realm name. (2) In a Tivoli environment, an operating system user ID that is associated with a Tivoli administrator.

principal password. In Kerberos, the password that corresponds to the principal name. This password is used to authenticate services and users to each other.

print file document. A Printing Systems Manager (PSM) object that represents text or data to be printed by a job. Contrast with print resource document.

Printing Systems Manager (PSM). An IBM licensed program that applies print administration and management technology to a cross-platform, client/server print system. PSM provides a set of (a) printing functions for submitting and controlling print jobs and (b) systems management and operator functions to control print spoolers and print supervisors. PSM is based on the Palladium distributed print system.

print resource document. A Printing Systems Manager (PSM) object that represents a resource, such as graphics or fonts, used by a job to print a print file document.

Print Services Facility™ (PSF) for AIX. An IBM licensed printer driver program that produces printer commands from the data sent to it.

private key. In computer security, a key that is known only to its owner. Contrast with public key. See public key cryptography.

profile. In a Tivoli environment, a container for application-specific information about a particular type of resource. A Tivoli application specifies the template for its profiles; the template includes information about the resources that can be managed by that Tivoli application.

A profile is created in the context of a profile manager; the profile manager links a profile to the Tivoli resource (for example, a managed node) that uses the information contained in the profile. A profile does not have any direct subscribers.

profile manager. In a Tivoli environment, a container for profiles that links the profiles to a set of resources, called “subscribers.” A profile manager can contain (a) profiles of multiple types or (b) multiple profiles of the same type. Tivoli administrators use profile managers to organize and distribute profiles. A profile manager is created in the context of a policy region and is a managed resource in a policy region. See subscription list.

program-to-program interface (PPI). In Tivoli NetView for OS/390, a facility that allows user programs to send data buffers to or receive data buffers from other user programs. It also allows system and application programs to send alerts to the NetView hardware monitor.

prototype profile. In a Tivoli environment, a model profile from which a Tivoli administrator can create other profiles, often by cloning the prototype profile.

proxy endpoint. In Tivoli Distributed Monitoring, a representation for an entity (such as a network device or a host) that functions as a subscriber for Tivoli Distributed Monitoring profiles. A Tivoli administrator associates each proxy endpoint with a managed node; several proxy endpoints can be associated with a single managed node.

PSCP. See presentation services command processor.

PSF. Print Services Facility. See Print Services Facility for AIX.

PSM. See Printing Systems Manager.

public key. In computer security, a key that is made available to everyone. Contrast with private key. See public key cryptography.

public key cryptography. In computer security, cryptography in which public keys and private keys are used for encryption and decryption.

pull. A network operation that initiates an action by requesting the action from a resource. Contrast with push.

push. A network operation that sends information to resources. Contrast with pull.

Q

query. In a Tivoli environment, a combination of statements that are used to search the configuration repository for systems that meet certain criteria.

query library. In a Tivoli environment, a facility that provides a way to create and manage Tivoli queries.

R

RACF. See Resource Access Control Facility.

RACF secured sign-on. In the Resource Access Control Facility (RACF), a function that enables workstations and other clients to sign on to the host and communicate in a secure way without having to send RACF passwords across the network. See PassTicket and PassTicket application key.

RDBMS. See relational database management system.

RDBMS Interface Module (RIM). In the Tivoli Management Framework, the module in the distributed object database that contains information about the installation of the relational database management system (RDBMS).

real object. In the NetView Graphic Monitor Facility, an object that represents an actual resource. See aggregate object.

real resource. (1) In VTAM, a resource identified by its real name and its real network identifier. (2) In the NetView Graphic Monitor Facility, an individual network resource represented by a real object.

recommended action. The procedures that Tivoli NetView for OS/390 recommends for determining and correcting the causes of network problems.

recording filter. In Tivoli NetView for OS/390, the function that determines which events, statistics, and alerts are stored in a database.

reference implementation. An implementation by which other implementations are judged for conformance to a standard or are tested for interoperability.

reference model. In the context of Tivoli software, the model configuration for a system or set of systems that is used to maintain consistent configurations in a distributed environment. In Tivoli Inventory, reference models are created in the configuration repository.

registered name. In a Tivoli environment, the name by which a particular resource is registered with the name registry when it is created.

registration file. See application registration file, field registration file, local registration file, and symbol registration file.

regular command. See immediate command.

relation. (1) In a relational database, a set of entity occurrences that have the same attributes. (T) (2) The comparison of two expressions to see if the value of one is equal to, less than, or greater than the value of the other. (3) In a relational database, a table that identifies entities and their attributes.

relational database. A database in which the data are organized and accessed according to relations. (T)

relational database management system (RDBMS). A collection of hardware and software that organizes and provides access to a relational database.

relative path. A path that begins with the working directory. Contrast with absolute path.

remote distribution. In a Tivoli environment, a distribution to target machines in a connected Tivoli Management Region.

remote host. Any host on a network except the host at which a particular operator is working.

Remote Operations Service (ROPS). In Communications Server, an application program on a client workstation that processes commands that are issued by Tivoli NetView for OS/390 through the Service Point Application (SPA) Router, thus enabling Tivoli NetView for OS/390 to manage distributed networks and application programs.

remote procedure call (RPC). (1) A facility that a client uses to request the execution of a procedure call from a server. This facility includes a library of procedures and an external data representation. (2) A client request to a service provider located in another node.

repeater. (1) A node of a local area network; a device that regenerates signals in order to extend the range of transmission between data stations or to interconnect two branches. (T) (2) See repeater site.

repeater range. In a Tivoli environment, the Tivoli clients that receive data from the repeater site.

repeater site. In a Tivoli Management Region, a managed node that is configured with the MDist feature. A repeater site receives a single copy of data and distributes it to the next tier of clients.

requester. See client.

Request for Comments (RFC). In Internet communications, the document series that describes a

part of the Internet suite of protocols and related experiments. All Internet standards are documented as RFCs.

resource. (1) Any facility of a computing system or operating system required by a job or task, and including main storage, input/output devices, the processing unit, data sets, and control or processing programs. (2) In Tivoli NetView for OS/390, any hardware or software that provides function to the network. (3) See managed resource.

Resource Access Control Facility (RACF). An IBM licensed program that provides for access control by identifying and verifying the users of the system, by authorizing access to protected resources, by logging the detected unauthorized attempts to enter the system, and by logging the detected accesses to protected resources.

resource label. In the NetView Graphic Monitor Facility, the textual information that identifies a particular aggregate or real resource. The resource label is displayed next to the resource symbol and cannot be changed by the network operator.

resource level. In Tivoli NetView for OS/390, the hierarchical position of a device (and the software contained within it) in a data processing system. For example, a first-level resource could be the communication controller, and the second-level resource could be the line connected to it.

resource manager. In Tivoli NetView for OS/390, an application program that manages specific network resources. Each resource manager is assigned a unique range of command indicators that specify the command support characteristics for the resources that it manages. The resource manager provides information to the NetView Graphic Monitor Facility (NGMF).

Resource Object Data Manager (RODM). In Tivoli NetView for OS/390, a component that operates as a cache manager and that supports automation applications. RODM provides an in-memory cache for maintaining real-time data in an address space that is accessible by multiple applications.

resource resolution table (RRT). In NetView Performance Monitor (NPM), a table that contains the names of the network resources for which data is to be collected. The NPM RRT corresponds with an NCP and is built by NPMGEN from an NCP Stage I and an NCP RRT.

resource status collector. In Tivoli NetView for OS/390, a function that collects status information on monitored resources and forwards this information to the resource status manager.

resource status manager. The part of the NetView Graphic Monitor Facility that maintains a database of SNA resource status information and that forwards this information to all attached server workstations.

resource symbol. In the NetView Graphic Monitor Facility, a geometric shape (such as a line, square, or octagon) that represents a particular kind of resource and indicates whether that resource is one resource or a composite of a group of resources.

resource type. (1) In a Tivoli environment, one of the properties of a managed resource. Resource types are defined in the default policy for a policy region. (2) In Tivoli NetView for OS/390, one of the three elements, which also include data type and display type, that are used to describe the organization of panels. Resource types in one category include central processing unit, channel, control unit, and I/O device; and in another category, they include communication controller, adapter, link, cluster controller, and terminal.

response level. See alarm level.

response time. (1) The elapsed time between the end of an inquiry or demand on a computer system and the beginning of the response; for example, the length of time between an indication of the end of an inquiry and the display of the first character of the response at a user terminal. (I) (A) (2) For response time monitoring, the time from the activation of a transaction until a response is received, according to the response time definition coded in the performance class.

response time monitor (RTM). A feature available with certain hardware devices to allow measurement of response times, which may be collected and displayed by Tivoli NetView for OS/390.

review file. In the NetView Performance Monitor (NPM), a VSAM key-sequenced data set (KSDS) containing data collected and recorded as a result of a network start display command or start monitor command.

RFC. See Request for Comments.

RIM. See RDBMS Interface Module.

RIM repository. See configuration repository.

RODM. See Resource Object Data Manager.

RODM-based view. In the NetView Graphic Monitor Facility (NGMF), a view that is predefined or dynamically built based on definitions in RODM. Examples of a RODM-based view are network views, exception views, configuration views, and more-detail views.

RODM resource. In the context of NetView Graphic Monitor Facility (NGMF) views, an object created in RODM to represent a resource. These objects can be created by loader files, user applications, or by the SNA topology manager.

role. See authorization role.

root directory. The highest level directory in a hierarchical file system.

root user. In the UNIX operating system, a user who has superuser authority.

ROPS. See Remote Operations Service.

router. (1) A computer that determines the path of network traffic flow. The path selection is made from several paths based on information obtained from specific protocols, algorithms that attempt to identify the shortest or best path, and other criteria such as metrics or protocol-specific destination addresses. (2) An attaching device that connects two LAN segments, which use similar or different architectures, at the reference model network layer. (3) In OSI terminology, a function that determines a path by which an entity can be reached. (4) Contrast with bridge.

RPC. See remote procedure call.

RRT. See resource resolution table.

RS/6000. A family of workstations and servers based on IBM's POWER architecture. They are primarily designed for running multiuser numerical computing applications that use the AIX operating system.

RTM. See response time monitor.

rule. In the Tivoli Enterprise Console, a set of one or more logical statements that enable the event server to recognize relationships among events (event correlation) and to execute automated responses accordingly.

rule base. In the Tivoli Enterprise Console, a set of rules and the event class definitions for which the rules are written. The Tivoli Enterprise Console uses the rule base in managing events. An organization can create many rule bases, with each rule base fulfilling a different set of needs for network computing management.

S

SAF. See System Authorization Facility.

scalable. Pertaining to the capability of a system to adapt readily to a greater or lesser intensity of use, volume, or demand. For example, a scalable system can efficiently adapt to work with larger or smaller networks performing tasks of varying complexity.

scanner. In a Tivoli environment, the software installed on each PC managed node that is to be scanned by Tivoli Inventory.

scheduler. A computer program designed to perform functions such as scheduling, initiation, and termination of jobs. (A)

schema. The set of statements, expressed in a data definition language, that completely describe the structure of a database.

scope check. In Tivoli NetView for OS/390, the process of verifying that an operator is authorized to issue a particular command. Contrast with span check. See scope of command authorization.

scope of command authorization. In Tivoli NetView for OS/390, the level of access authority that a system programmer or system administrator grants to a network operator to use various commands. See scope check.

script. (1) A computer program that is interpreted. (2) See shell script.

script stub. A placeholder for a particular shell script. For example, the Tivoli Module Builder generates a script stub (using a skeleton file) if a developer does not provide the script for implementing a particular task or monitor when defining the task or monitor; the script stub then displays a message that the script executed successfully and displays any variables that were passed to the script.

search path. See path.

seat. A slang term that refers to the number of licensed users of a software product, which is the same as the number of installations of the product. For example, if there were 100 Lotus Notes™ seats, there would be 100 licensed users of Lotus Notes (or 100 installations of Lotus Notes).

secondary database. One of two databases provided by Tivoli NetView for OS/390 for recording data. It provides backup or a temporary storage alternative to the primary database. See primary database.

security group. In a Tivoli environment, a group of managed resources over which a Tivoli administrator is granted authority. Examples of a security group include a policy region and the administrator collection.

security management. The Tivoli management discipline that addresses the organization's ability to control access to applications and data that are critical to its success. See availability management, deployment management, and operations and administration.

seed file. In Tivoli NetView, a file that contains a list of nodes within an Administrative Domain, which the automatic discovery function uses to accelerate the generation of the network topology map.

segment. (1) A portion of a computer program that may be executed without the entire computer program being resident in main storage. (T) (2) A group of display elements. (3) A section of cable between components or devices. A segment may consist of a single patch cable, several patch cables that are

connected, or a combination of building cable and patch cables that are connected. (4) In the Enhanced X-Windows Toolkit, one or more lines that are drawn but not necessarily connected at the endpoints. (5) In LANs or WANs, a subset of nodes in a network or subnet that are connected by a common physical medium.

senior role. See authorization role.

server. A functional unit that provides services to one or more clients over a network. Examples include a file server, a print server, and a mail server.

server workstation. In the NetView Graphic Monitor Facility, a workstation with the graphic data server. This workstation uses the graphic monitor and the view administrator for administrative functions. The server workstation sends status information to client workstations over an LU 6.2 session.

Service Level Reporter (SLR). A licensed program that generates management reports from data sets such as System Management Facility (SMF) files.

service point (SP™). An entry point that supports applications providing network management for resources that are not under its direct control as an entry point. Each resource is either under the direct control of another entry point or not under the direct control of any entry point. A service point accessing these resources is not required to use SNA sessions (unlike a focal point).

Service Point Application Router. In Communications Server, software that receives commands issued from Tivoli NetView for OS/390 and sends these commands to an application program, called the Remote Operations Service (ROPS), to be processed on a client workstation.

service point command facility (SPCF). A program or function that exchanges data and control between the network operator, the link connection component manager (LCCM), and the link connection subsystem manager (LCSM).

service point command service (SPCS). In Tivoli NetView for OS/390, an extension of the command facility that allows the host processor to communicate with a service point by using the communication network management (CNM) interface.

session data. Session awareness data, session trace data, and session response time data that Tivoli NetView for OS/390 collects.

session manager (SM). A product, such as NetView Access Services, that allows a user at a terminal to log on to multiple applications concurrently.

session monitor. In Tivoli NetView for OS/390, the component that collects and correlates session-related data and provides online access to this information. Contrast with hardware monitor.

session setup failure notification (SSFN). In Tivoli NetView for OS/390, session awareness data that is provided when there is a failure. It identifies the system services control point (SSCP) that detects the error, the SSCPs that are involved, and the names of the session partners affected.

session statistics file. In the NetView Performance Monitor (NPM), an online VSAM key-sequenced data set (KSDS) used for storing session data.

session trace. In Tivoli NetView for OS/390, the function that collects session trace data for sessions involving specified resource types or involving a specific resource.

session trace data. Data, relating to sessions, that is collected by Tivoli NetView for OS/390 whenever a session trace is started and that consists of session activation parameters, VTAM path information unit (PIU) data, and NCP data.

severity level. In the Tivoli Enterprise Console, a classification for an event that indicates its degree of severity. Severity levels can be modified by a user or a Tivoli Enterprise Console rule. The predefined severity levels, in order of descending severity, include: fatal, critical, warning, minor, harmless, and unknown.

shared application program. In Tivoli NetView, an application program that serves multiple action requests; however, only one instance of the application program can run in a given graphical user interface (GUI).

shared submap. In Tivoli NetView, a submap on which multiple application programs manage objects on the application plane. Shared submaps allow application programs to cooperatively contribute information to the same submap. Contrast with exclusive submap.

shell. A software interface between a user and the operating system of a computer. Shell programs interpret commands and user interactions on devices such as keyboards, pointing devices, and touch-sensitive screens and communicate them to the operating system. Shells simplify user interactions by eliminating the user's concern with operating system requirements. A computer may have several layers of shells for various levels of user interaction.

shell procedure. See shell script.

shell prompt. In the UNIX operating system, the character string on the command line indicating that the system can accept a command (typically the $ character).

shell script. In the UNIX operating system, a series of commands, combined in a file, that carry out a particular

function when the file is run or when the file is specified as a value to the SH command.

show cause. The reason code in the record maintenance statistics (RECMS) that indicates to VTAM or to Tivoli NetView for OS/390 the threshold that was exceeded and whether the threshold has been dynamically altered.

SIA. System Information Agent. See Tivoli Distributed Monitoring, the product that replaces the System Information Agent.

signal. In computer software, a message that is sent to a process to change its behavior based on the value sent to it.

signature. In computer software, the name of an operation and its parameters.

simple connection. In Tivoli NetView, the representation of connectivity as seen from one endpoint of a connection.

Simple Network Management Protocol (SNMP). In the Internet suite of protocols, a network management protocol that is used to monitor routers and attached networks. SNMP is an application layer protocol. Information on devices managed is defined and stored in the application's Management Information Base (MIB).

singular filter. A filter that identifies a host, subnet, or all hosts with a single expression.

skeleton file. A program template that the Tivoli Module Builder uses to generate any text-based file, including scripts, Java or C source files, build files, and help text. A skeleton file includes substitution variables that are replaced at run time. The values for these variables originate from user-defined variables or values specified in a component description file (CDF) or a global description file (GDF) file.

SLR. See Service Level Reporter.

SMF. See System Management Facility.

SMIT. See System Management Interface Tool.

SMS. See Storage Management Subsystem.

snapshot. In Tivoli NetView, a copy of a map that reflects the topology and status of the map's nodes and links at a given moment in time.

SNATM. See SNA topology manager.

SNA topology manager (SNATM). In Tivoli NetView for OS/390, a component that dynamically collects status and topology data into the Resource Object Data Manager (RODM) for display by the NetView Graphic Monitor Facility (NGMF). SNATM includes the function formerly provided by the APPN Topology and Accounting Manager (APPNTAM) feature of NetView for MVS V2R4.

SNMP. See Simple Network Management Protocol.

Software Installer for OS/2. An OS/2-based tool that is used to install workstation functions such as the NetView Graphic Monitor Facility.

software management. See change management.

source host. In Tivoli Software Distribution, the managed node on which the files and directories referenced in a file package reside.

span. In Tivoli NetView for OS/390, a user-defined group of network resources within a single domain. Spans provide a level of security by allowing the system administrator to define (a) the resources to which an operator can issue commands, (b) the views of resources that an operator can display, and (c) the resources in a view that an operator is allowed to see (an operator may not be authorized to see all the resources in a particular view). See span check.

span check. In Tivoli NetView for OS/390, the process of verifying that an operator is authorized to perform actions on a network resource, a NetView Graphic Monitor Facility (NGMF) view, or a resource within a view. Contrast with scope check.

SPA Router. See Service Point Application Router.

SPCF. See service point command facility.

SQL. A programming language that is used to define and manipulate data in a relational database.

SRF. See symbol registration file.

SSFN. See session setup failure notification.

SSI. See subsystem interface.

stage. In Tivoli NetView for OS/390, a program that processes messages in a NetView pipeline. Stages send messages to each other serially.

statistics record. In Tivoli NetView for OS/390, a resource-generated database record that contains various statistics about a resource.

status monitor. In Tivoli NetView for OS/390, a component that collects and summarizes information on the status of resources defined in a VTAM domain.

Storage Management Subsystem (SMS). A DFSMS/MVS facility that is used to automate and centralize the management of storage. Using SMS, a storage administrator describes data allocation characteristics, performance and availability goals, backup and retention requirements, and storage requirements to the system.

Structured Query Language. See SQL.

subagent. In the Simple Network Management Protocol (SNMP), something that provides an extension to the utility provided by the SNMP agent.

submap. In Tivoli NetView, a particular view of some aspect of a network that displays symbols representing objects. The application program that creates a submap determines what part of the network the submap displays.

submap pane. The area of a submap window in which the submap is displayed.

submap stack. In Tivoli NetView, a component of the graphical user interface shown on the left side of each submap window. The submap stack represents the navigational path used to reach the particular submap, and it can be used to select a previously viewed submap.

submap window. In Tivoli NetView, the graphical component that contains a menu bar, a submap viewing area, a status line, and a button box. A user can display multiple submap windows of an open map and an open snapshot at any given time. See primary window.

subnetwork. Any group of nodes that have a set of common characteristics, such as the same network ID.

subscriber. In a Tivoli environment, a Tivoli client, a profile manager, or any endpoint type (for example, a PC managed node or a proxy endpoint) that is subscribed to a profile manager. Although profiles are distributed to a subscriber, the subscriber may or may not be the final destination of the profile distribution.

subscription. In a Tivoli environment, the process of identifying the subscribers to which profiles will be distributed.

subscription list. In a Tivoli environment, a list that identifies the subscribers to a profile manager. Including a profile manager on a subscription list (in effect, a list within a list) is a way of subscribing several resources simultaneously rather than adding each one individually. In Tivoli Plus modules, a profile manager functions as a subscription list.

subsystem interface (SSI). The MVS interface by which routines (IBM-, vendor-, or installation-written) request services of, or pass information to, subsystems. The SSI is used by Tivoli NetView for OS/390 to receive system messages and enter system commands (when used with extended MCS consoles, it is used to receive commands, not messages), and to communicate with other instances of Tivoli NetView for OS/390.

super role. See authorization role.

superuser authority. In the UNIX operating system, the unrestricted authority to access and modify any part of the operating system, usually associated with the user who manages the system. See root user.

suppression character. In Tivoli NetView for OS/390, a user-defined character that is coded at the beginning of a command list statement or a command to prevent the statement or command from appearing on the operator's terminal screen or in the network log.

symbol. In Tivoli NetView, a picture or an icon on a submap that represents an object (a network resource or an application). Each symbol belongs to a class, represented by the symbol's shape, and to a subclass, represented by the design within the shape. The symbol reflects characteristics of the object it represents, such as its status; it also has characteristics of its own, such as behavior.

symbol registration file (SRF). In Tivoli NetView, a file used to define symbol classes and subclasses.

synchronous monitor. In Tivoli Distributed Monitoring, a monitor that monitors resources on a periodic basis (most monitors are synchronous). Contrast with asynchronous monitor.

sysplex. A set of MVS or OS/390 systems communicating and cooperating with each other through certain multisystem hardware components and software services to process customer workloads. This term is derived from "system complex."

System Authorization Facility (SAF). An interface defined by MVS that enables programs to use system authorization services in order to protect access to resources such as data sets and MVS commands. The IBM Resource Access Control Facility (RACF) is a product that uses the SAF interface.

system configuration. A process that specifies the devices and programs that form a particular data processing system.

System Information Agent (SIA). See Tivoli Distributed Monitoring, the product that replaces the System Information Agent.

System Management Facility (SMF). A standard feature of OS/390 that collects and records a variety of system and job-related information.

System Management Interface Tool (SMIT). An interface tool of the AIX operating system for installing, maintaining, configuring, and diagnosing tasks.

systems management. (1) Functions in the application layer related to the management of Open Systems Interconnection (OSI) resources and their status across all layers of the OSI architecture. (I) (2) The tasks involved in maintaining computer and communication systems, for example: changing

configuration, identifying faults, securing access, accounting for resource usage, and analyzing performance.

T

TACF. See Tivoli Access Control Facility.

TACF database. In Tivoli Security Management, a database that contains the customized rules that the authorization daemon for the Tivoli Access Control Facility (TACF) uses to allow or to deny resource accesses in the UNIX environment.

TACF lookaside database. In Tivoli Security Management, a database that provides ID-to-name resolution, thereby enabling the Tivoli Access Control Facility (TACF) to convert UNIX IDs (user IDs, group IDs, IP addresses, and port numbers) to names at run time.

TAF. See terminal access facility.

TAP. See Telocator Alphanumeric Protocol.

target. See endpoint and execution target.

target host. See endpoint.

task. (1) In a multiprogramming or multiprocessing environment, one or more sequences of instructions treated by a control program as an element of work to be accomplished by a computer. (I) (A) (2) In a Tivoli environment, the definition of an action that must be routinely performed on various managed nodes throughout the network. A task defines the executables to be run when the task is executed, the authorization role required to execute the task, and the user or group name under which the task will execute.

task endpoint. See endpoint.

task library. In a Tivoli environment, a container in which a Tivoli administrator can create and store tasks and jobs.

Task Library Language (TLL). In a Tivoli environment, a programming language used to define a task library. The TLL definition can be used to copy a task library from one installation to another. The TLL also allows the arguments for each task to be described such that graphical user interface (GUI) tools can interpret them and present an interface for operators who want to create the tasks.

TCP/IP. See Transmission Control Protocol/Internet Protocol.

Telocator Alphanumeric Protocol (TAP). An industry standard protocol for the input of paging requests.

terminal access facility (TAF). In Tivoli NetView for OS/390, a facility that allows a network operator to control a number of subsystems. In a full-screen or operator control session, operators can control any combination of such subsystems simultaneously.

threshold. In software products, a value that defines a limit for a monitored condition. The monitored condition, the significance of the limit, and the particular software product's response when the monitored condition reaches the specified threshold vary widely according to product.

throttle. In Tivoli NetView, a condition defined in the filter table and used to regulate the flow of traps.

time to live (TTL). A technique used by best-effort delivery protocols to inhibit endlessly looping packets. The packet is discarded if the TTL counter reaches 0.

Tivoli Access Control Facility (TACF). In Tivoli Security Management, an object-oriented security system that runs on UNIX-based operating systems and provides security functions that are not available on UNIX (such as an access rule database, an audit log, and administration tools). TACF is invoked immediately after the operating system has completed its initialization, and it places hooks in system services that should be protected, thereby enabling control to be passed to TACF before the services are performed.

Tivoli administrator. In a Tivoli environment, a system administrator who has been authorized to perform systems management tasks and manage policy regions in one or more networks. Each Tivoli administrator is represented by an icon on the Tivoli desktop.

Tivoli Application Development Environment. A Tivoli toolkit that contains the complete application programming interface (API) for the Tivoli Management Framework. This toolkit enables customers and Tivoli Partners to develop their own applications for the Tivoli environment.

Tivoli Application Extension Facility. A Tivoli toolkit that enables customers to extend the capabilities of Tivoli applications. For example, customers can add fields to a dialog, create custom attributes and methods for application resources, or create custom icons and bitmaps.

Tivoli client. A client of a Tivoli server. See TMR client and TMR server.

Tivoli Cross-Site®. The integrated suite of Tivoli products for managing an e-commerce environment to ensure that Web resources are secure and available and to enable applications and information to be distributed and maintained across the extended enterprise.

Tivoli Decision Support. A Tivoli product that consolidates, transforms, and presents IT data in many different views, enabling an enterprise to gain insight

into patterns and relationships among the data and to make critical business decisions based on this data.

Tivoli desktop. In a Tivoli environment, the desktop that system administrators use to manage their network computing environment.

Tivoli Developer Kit. See Tivoli Module Designer.

Tivoli Distributed Monitoring. A Tivoli product that monitors system resources, initiates any necessary corrective actions, and informs system administrators of potential problems. Tivoli Distributed Monitoring consists of a group of monitors that are installed on each managed node that is to be monitored. It resolves some events on its own and may send others to the Tivoli Enterprise Console.

Tivoli Enterprise Console. A Tivoli product that collects, processes, and automatically initiates corrective actions for system, application, network, and database events; it is the central control point for events from all sources. The Tivoli Enterprise Console provides a centralized, global view of the network computing environment; it uses distributed event monitors to collect information, a central event server to process information, and distributed event consoles to present information to system administrators.

Tivoli Enterprise software. The integrated suite of Tivoli products for systems management in a large organization. These products enable system administrators to manage their network computing enterprise according to the disciplines of availability management, deployment management, operations and administration, security management, and service-level management. This suite includes Tivoli Global Enterprise Manager, Tivoli NetView for OS/390, and Tivoli Decision Support.

Tivoli environment. The Tivoli applications, based upon the Tivoli Management Framework, that are installed at a specific customer location and that address network computing management issues across many platforms. In a Tivoli environment, a system administrator can distribute software, manage user configurations, change access privileges, automate operations, monitor resources, and schedule jobs.

Tivoli Event Integration Facility. A Tivoli toolkit that provides a simple application programming interface (API) to enable customers and Tivoli Partners to develop new event adapters that can forward events to the Tivoli Enterprise Console. A customer can also translate events from third-party or in-house applications.

Tivoli GEM. See Tivoli Global Enterprise Manager.

Tivoli GEM module. In a Tivoli environment, a management module that enables a particular application or business system to be managed by the Tivoli Global Enterprise Manager (Tivoli GEM).

Tivoli Global Enterprise Manager (Tivoli GEM). A Tivoli product that allows system administrators to graphically monitor, control, and configure applications residing in distributed and host (S/390®) environments and to use the concept of business systems management to organize related components, thereby providing a business perspective for management decisions. Tivoli Global Enterprise Manager gives information technology staff a logical view of the computing environment; this view shows, at a glance, the status of the multiple applications that comprise the enterprise's business system, including application components, the relationships among and between components, and the flow of data between the applications. By providing this view from a business perspective, Tivoli Global Enterprise Manager enables system administrators to quickly make determinations about the business impact of any component failure. Addressing technology problems from the business perspective greatly improves the effectiveness of system administrators and provides a higher level of service to users.

Tivoli install image. In a Tivoli environment, a file that resides on a CD or in a file system and contains a Tivoli product to be installed. A Tivoli install image can be used to install the Tivoli Management Framework or to install an application onto the Framework for the first time. A single CD often includes both a Tivoli install image and a Tivoli upgrade image, and it may include Tivoli install images for more than one application. Contrast with Tivoli upgrade image.

Tivoli Inventory. A Tivoli product that enables system administrators to gather hardware and software information for a network computing environment. It scans the managed resources and stores inventory information in the configuration repository.

Tivoli IT Director. A Tivoli product for systems management in a small or medium organization. It is not sold directly by Tivoli Systems Inc. but rather through a Tivoli authorized reseller.

Tivoli LAN Access. A Tivoli product that enables system administrators to extend existing LAN management tools by integrating them with the Tivoli suite of products.

Tivoli management agent. In the Tivoli environment, an agent that securely performs administrative operations.

Tivoli Management Framework. The base software that is required to run the applications in the Tivoli product suite. This software infrastructure enables the integration of systems management applications from Tivoli Systems Inc. and the Tivoli Partners. The Tivoli Management Framework includes the following:

- Object request broker (`oserv`)
- Distributed object database

- Basic administration functions
- Basic application services
- Basic desktop services such as the graphical user interface

In a Tivoli environment, the Tivoli Management Framework is installed on every client and server; however, the TMR server is the only server that holds the full object database.

Tivoli management gateway. In the Tivoli environment, a system that enables bidirectional communication with Tivoli Management Agents.

Tivoli Management Region (TMR). In a Tivoli environment, a Tivoli server and the set of clients that it serves. An organization can have more than one TMR. A TMR addresses the physical connectivity of resources whereas a policy region addresses the logical organization of resources.

Tivoli management software. The overall descriptor for software from Tivoli Systems Inc., which includes Tivoli Enterprise software (for systems management in a large organization), Tivoli IT Director (for systems management in a small or medium organization), and Tivoli Cross-Site (for the management of e-commerce systems). Tivoli management software enables organizations to centrally manage their computing resources (including the critical applications that drive business performance and profits) in a simple and straightforward manner.

Tivoli Manager. Tivoli management software that manages specific vendor systems, networks, applications, or databases.

Tivoli Module Builder (TMB). A Tivoli product that enables developers to create a special type of file, called a management module, for managing an application or business system with Tivoli management software. Management modules include Tivoli GEM modules and Tivoli Plus modules. The Tivoli Module Builder provides tools (such as the Tivoli Module Designer) and templates for describing the management characteristics of an application or business system and for building this information (together with the scripts, programs, and files that are required to implement the management function) into a Tivoli install image or an application management package. The Tivoli Module Builder uses file types defined in the Application Management Specification (AMS).

Tivoli Module Designer (TMD). A Tivoli tool that enables developers to describe the management characteristics of an application or business system and that generates the application description files and application management packages that the Tivoli management software uses to manage applications and business systems. The Tivoli Module Designer replaces the Tivoli Developer Kit.

Tivoli NetView. A Tivoli product that enables distributed network management across multiple operating systems and protocols. Unlike Tivoli NetView for OS/390, Tivoli NetView does not provide centralized management from an OS/390 host.

Tivoli NetView for OS/390. A Tivoli product that enables centralized systems and network management from an OS/390 environment. Through its MultiSystem Manager component, Tivoli NetView for OS/390 enables management of distributed resources, such as Internet Protocol (IP) resources, NetWare resources, asynchronous transfer mode (ATM) resources, and others. Contrast with Tivoli NetView.

Tivoli NetWare repeater (TNWR). In a Tivoli environment, a server application that is installed on a Novell NetWare server and that maintains a list of available clients for the server. The Tivoli NetWare repeater works with the NetWare managed site to perform profile distribution.

Tivoli Partner Association. A partnership program that is led by Tivoli Systems Inc. for business, industry, and product partners. The Tivoli Partner Association provides programs and benefits for business partners (including systems integrators, outsourcers, and resellers) to sell Tivoli Enterprise and IT Director products. Industry and product partners collaborate with Tivoli Systems Inc. in creating hardware and software products that are Tivoli Ready.

Tivoli Plus module. In a Tivoli environment, a management module that has been certified by the Tivoli Partner Association and that enables a specific vendor application to be managed by Tivoli management software. To be certified by the Tivoli Partner Association, the Tivoli Plus module must include certain features such as enablement for the Tivoli Global Enterprise Manager (Tivoli GEM).

Tivoli Ready. Pertaining to a product that has passed rigorous product certification testing by Tivoli Systems Inc. to ensure that the product delivers turnkey (or ″out-of-the-box″) integration with Tivoli management software. A product that has passed this certification testing carries the Tivoli Ready logo.

Tivoli Remote Control. A Tivoli product that enables a Tivoli administrator to control mouse and keyboard operations on an NT managed node or a PC managed node.

Tivoli Remote Execution Service. A service that enables a Tivoli environment to perform remote operations on machines. These operations include: remotely installing clients, connecting Tivoli Management Regions (TMRs), and starting `oserv` from a remote machine.

Tivoli Security Management. Tivoli Enterprise software that enables the consistent definition,

implementation, and enforcement of security policy in a network computing environment.

Tivoli server. The server that holds or references the complete set of Tivoli software, including the full object database. See Tivoli client, TMR client, and TMR server.

Tivoli Service Desk for OS/390. A Tivoli product that is an integrated set of tools, services, and interfaces for automating and customizing a organization's IT service and support operation in an OS/390 environment. It provides a structure that supports the gathering, organizing, locating, and reporting of information related to problem, change, and asset management.

Tivoli Software Distribution. A Tivoli product that automates software distribution to clients and servers in a network computing environment. An organization can use this product to install and update applications and software in a coordinated, consistent manner across a network. Tivoli Software Distribution creates file packages and distributes them to predefined subscribers.

Tivoli upgrade image. In a Tivoli environment, a file that resides on a CD or in a file system and contains updates for a Tivoli product. A Tivoli upgrade image contains only the files that have changed since the previous product release, with the scripts and commands that are needed for installing the new files and configuring the database. Contrast with Tivoli install image.

Tivoli User Administration. A Tivoli product that provides a graphical user interface (GUI) for centralized management of user and group accounts. It offers efficient, automated management of user and system configuration parameters, secure delegation of administrative tasks, and centralized control of all user and group accounts in a network computing environment.

Tivoli UserLink. A Tivoli product that provides IP address synchronization between a PC agent and its associated PC managed node using the Dynamic Host Configuration Protocol (DHCP). Tivoli UserLink also enables a PC user to pull a file package to a Windows, Windows 95, or Windows NT workstation.

TLL. See Task Library Language.

TMB. See Tivoli Module Builder.

TMD. See Tivoli Module Designer.

TME 10. See Tivoli Enterprise software.

TMR. See Tivoli Management Region.

TMR client. In a Tivoli environment, any computer—except the TMR server—on which the Tivoli Management Framework is installed. The `oserv` daemon runs on the TMR client, and the TMR client maintains a local object database. See Tivoli client and Tivoli server.

TMR server. A Tivoli server for a specific Tivoli Management Region (TMR). See Tivoli client and TMR client.

TNWR. See Tivoli NetWare repeater.

toggle button. In the AIXwindows Toolkit and the Enhanced X-Windows Toolkit, a graphical object that simulates a toggle switch; it switches sequentially from one optional state to another.

tool palette. In Tivoli NetView, a component of the graphical user interface (GUI) that enables the network operator to open application program instances by using the mouse to drag and drop the icons that represent the application program.

topology. In communications, the physical or logical arrangement of nodes in a network, especially the relationships among nodes and the links between them.

topology console. In the Tivoli Global Enterprise Manager and Tivoli NetView for OS/390, a Java-based graphical user interface that displays business system information from the topology server. The topology console displays each component as a separate icon or shape and draws lines between icons to denote links. It then uses color to indicate the status of each component and of the business system as a whole. As the topology server receives configuration and status updates for the business system, it updates the topology console. Therefore, the topology console always displays the real-time configuration and status of the business system.

topology database. See local topology database and network topology database.

topology database update (TDU). A message about a new or changed link or node that is broadcast among APPN network nodes to maintain the network topology database, which is fully replicated in each network node. A TDU contains information that identifies the following:

- The sending node
- The node and link characteristics of various resources in the network
- The sequence number of the most recent update for each of the resources described.

topology server. In Tivoli Global Enterprise Manager and Tivoli NetView for OS/390, a server that interacts with instrumented applications in a business system and provides information for display on the topology console. The topology server receives heartbeat events from instrumented applications or components and determines the business system in which a component belongs. The topology server also queries instrumented applications for related applications and for the status of

its monitors. All of this information is used to create and maintain a view of each business system's configuration and availability on the topology console.

trace. A record of the execution of a computer program. It exhibits the sequences in which the instructions were executed. (A)

transaction. A specific set of input data that triggers execution of a specific process or job; a message destined for an application program.

transit time. See response time.

Transmission Control Protocol/Internet Protocol (TCP/IP). A set of communications protocols that support peer-to-peer connectivity functions for both local and wide area networks.

trap. In the Simple Network Management Protocol (SNMP), a message sent by a managed node (agent function) to a management station to report an exception condition.

triggered response. In a Tivoli environment, the action that is taken when a monitor reaches or exceeds a threshold.

trouble ticket. In Tivoli NetView, a record of a problem that has occurred. The trouble ticket becomes the formal vehicle to trace a problem from its occurrence to its resolution.

TTL. See time to live.

tuple. In a relational database, a part of a relation that uniquely describes an entity and its attribute. A tuple can be represented by one row of a relation table. (T)

U

UDP. See User Datagram Protocol.

underlying connection. In Tivoli NetView, the representation of lower-layer connectivity that is used by higher-layer connectivity. For example, the physical connection that transports data between two IP hosts is an underlying connection.

unmarshall. To copy data from a remote procedure call (RPC) packet. Stubs perform unmarshalling. Contrast with marshall.

upcall. In a Tivoli environment, a method invocation from an endpoint "up" to the gateway. Contrast with downcall.

User Datagram Protocol (UDP). In the Internet suite of protocols, a protocol that provides unreliable, connectionless datagram service. It enables an application program on one machine or process to send a datagram to an application program on another machine or process. UDP uses the Internet Protocol (IP) to deliver datagrams.

user login map. In a Tivoli environment, a mapping that associates a single user login name with a user account on a specified operating system. User login maps enable Tivoli administrators to log in to the Tivoli environment or perform operations within the Tivoli environment with a single user login name, regardless of the system that they are currently using.

user plane. In Tivoli NetView, the submap layer on which symbols of objects that are not managed by an application program are displayed. Symbols on the user plane are displayed with a shadow, which makes them appear higher than symbols on the application plane. See background plane.

user profile. (1) In computer security, a description of a user that includes such information as user ID, user name, password, access authority, and other attributes obtained at logon. (2) In Tivoli User Administration, a profile that is used to manage user accounts, including account information, home directories, startup files, and group membership.

user role. See authorization role.

using node. The NCP in the host's domain that reports a link error condition.

V

validation. The checking of data for correctness or for compliance with applicable standards, rules, and conventions. (A)

validation policy. In a Tivoli environment, policy that ensures that all resources in a policy region comply with the region's established policy. Validation policy prevents Tivoli administrators from creating or modifying resources that do not conform to the policy of the policy region in which the resources were created.

variable. (1) In programming languages, a language object that may take different values, one at a time. The values of a variable are usually restricted to a certain data type. (I) (2) A quantity that can assume any of a given set of values. (A) (3) A name used to represent a data item whose value can be changed while the program is running. (4) In the Simple Network Management Protocol (SNMP), a match of an object instance name with an associated value. (5) In the NetView command list language, a character string beginning with "&" that is coded in a command list and is assigned a value during execution of the command list.

verb. (1) In Tivoli NetView for OS/390, the first word of a NetView command that is delimited by a blank or a comma and that indicates what action is to be taken. (2) See LU 6.2 verb.

view administrator. The part of the NetView Graphic Monitor Facility that downloads the views created by the view preprocessor and that provides these views to the graphic data server.

viewing filter. In Tivoli NetView for OS/390, the function that allows a user to select the alert data to be displayed on a terminal. All other stored data is blocked.

view manager. In the NetView Graphic Monitor Facility, a facility that generates views according to Resource Object Data Manager (RODM) definitions and that provides status changes to the graphic data server.

view preprocessor. The part of the NetView Graphic Monitor Facility that creates unformatted views of SNA resources from the VTAM definition library (VTAMLST).

view preprocessor resource. An SNA subarea resource whose status is reported by the resource status manager and is stored in the graphic data server (GDS) databases when views containing the resource are downloaded.

Virtual Telecommunications Access Method (VTAM). An IBM licensed program that controls communication and the flow of data in an SNA network. It provides single-domain, multiple-domain, and interconnected network capability.

vital product data (VPD). Information that uniquely defines system, hardware, software, and microcode elements of a processing system.

VPD. See vital product data.

VTAM. See Virtual Telecommunications Access Method.

W

webmaster. The person who is ultimately responsible for managing and maintaining a particular Web site.

well-behaved application program. An application program that runs without disruption to the network.

widget. (1) In the AIX operating system, a graphic device that can receive input from the keyboard or mouse and can communicate with an application or with another widget by means of a callback. Every widget is a member of only one class and always has a window associated with it. (2) The fundamental data type of the Enhanced X-Windows Toolkit. (3) An object that provides a user-interface abstraction; for example, a Scrollbar widget. It is the combination of an Enhanced X-Windows window (or subwindow) and its associated semantics. A widget implements procedures through its widget class structure.

wildcard character. See pattern-matching character.

with-request. A Printing Systems Manager (PSM) document transfer method in which the client transfers documents directly to the server. This is the default transfer method. Contrast with dce-pipe-pull.

wizard. A dialog within an application that uses step-by-step instructions to guide a user through a specific task.

working directory. The directory that is currently in use by an operating system or application. If no path is specified, this is the directory to which data is written, from which data is deleted, or in which data is searched.

work space. (1) That portion of main storage that is used by a computer program for temporary storage of data. (I) (A) (2) In Tivoli NetView, a container for a set of event cards that meet certain criteria. See event filter.

wrap count. In Tivoli NetView for OS/390, the number of events that can be retained in the database for a specific resource or the number of alerts that are retained in the database.

X

XCF. See cross-system coupling facility.

X Window System. A software system, developed by the Massachusetts Institute of Technology, that enables the user of a display to concurrently use multiple application programs through different windows of the display. The application programs may execute on different computers.

Y

Year 2000 challenge. A term used especially by the computer industry to refer to the problems, challenges, and issues involved in preparing computer systems and applications for transition to, and operation in, the twenty-first century. For example, many computer systems and applications use two digits to represent the year (“97” rather than 1997). When these computer systems and applications encounter the digits “00” for the year 2000, they can misinterpret this to mean the year 1900 and can produce computing errors or fail to function. Although some systems and applications may not be affected until the eve of the new millennium (on 31 December 1999), many systems and applications that use future dates (such as expiration dates for credit cards) have already experienced Year 2000 problems. This problem could also affect such things as elevator

controls; household appliances such as VCRs and programmable coffee makers; heating, cooling, and security systems; telephone calls; driver's licenses; automated teller machines and bank vaults; and airline flight schedules.

Year 2000 ready. A product is Year 2000 ready if the product, when used in accordance with its associated documentation, is capable of correctly processing, providing, and/or receiving date data within and between the twentieth and twenty-first centuries, provided that all products (for example, hardware, software, and firmware) used with the product properly exchange accurate date data with it.

Y2K. See Year 2000 challenge.

Z

zombie process. In the UNIX operating system, a process that has been terminated but has not been cleaned up by its parent process. The existence of a large number of zombie processes could indicate an errant network daemon or application. Zombie processes are sometimes called "lingering terminated processes."

zoom. In a user interface, to progressively increase or decrease the size of a part of an image on a screen or in a window.

Numerics

4700 Support Facility. In Tivoli NetView for OS/390, a component that enables the monitoring and control of IBM 3600 and 4700 Finance Communication Systems. The 4700 Support Facility can record, analyze, and display performance and status data on IBM 3600 and 4700 Finance Communication Systems.

Index

E

F

G

H

I

K

L

M

N

O

P

R

S

T

U