Tivoli® NetView® for OS/390®

NetView Management Console User's Guide

Version 1 Release 4

GC31-8665-02

Tivoli® NetView® for OS/390®

NetView Management Console User's Guide

Version 1 Release 4

GC31-8665-02

Tivoli NetView for OS/390 NetView Management Console User's Guide

Trademarks

IBM, the IBM logo, Tivoli, the Tivoli logo, AIX, APPN, BookManager, DB2, MVS/ESA, NetView, OS/2, OS/390, Tivoli Enterprise Console, TME, VisualAge, VTAM and z/OS are trademarks or registered trademarks of International Business Machines Corporation or Tivoli Systems Inc. in the United States, other countries, or both.

Lotus is a registered trademark of Lotus Development Corporation.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Intel is a trademark of Intel Corporation in the United States, other countries, or both.

Other company, product, and service names may be trademarks or service marks of others.

Notices

References in this publication to Tivoli Systems or IBM products, programs, or services do not imply that they will be available in all countries in which Tivoli Systems or IBM operates. Any reference to these products, programs, or services is not intended to imply that only Tivoli Systems or IBM products, programs, or services can be used. Subject to valid intellectual property or other legally protectable right of Tivoli Systems or IBM, any functionally equivalent product, program, or service can be used instead of the referenced product, program, or service. The evaluation and verification of operation in conjunction with other products, except those expressly designated by Tivoli Systems or IBM, are the responsibility of the user. Tivoli Systems or IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, New York 10504-1785, U.S.A.

Programming Interfaces

This publication documents intended Programming Interfaces that allow the customer to write programs to obtain services of Tivoli NetView for OS/390.

Contents

Preface

This document provides overview information for the NetView® management console (NMC) interface to Tivoli® NetView for OS/390®. For more detailed information about specific interface functions, see the NetView management console online help system.

Who Should Read This Document

This document is intended for the network operators or system programmers responsible for operating the NetView management console (NMC) interface to Tivoli NetView for OS/390.

Prerequisite and Related Documents

To read about the new functions offered in this release, refer to the *Tivoli NetView for OS/390 Installation: Migration Guide*.

You can find additional product information on these Internet sites:

Table 1. Resource Address (URL)

IBM®	http://www.ibm.com/
Tivoli Systems	http://www.tivoli.com/
Tivoli NetView for OS/390	http://www.tivoli.com/nv390

The Tivoli NetView for OS/390 home page offers demonstrations of NetView, related products, and several free NetView applications you can download. These applications can help you with tasks such as:

- Getting statistics for your automation table and merging the statistics with a listing of the automation table
- Displaying the status of a JES job or cancelling a specified JES job
- Sending alerts to NetView using the program-to-program interface (PPI)
- Sending and receiving MVS commands using the PPI
- Sending TSO commands and receiving responses
- Making NetView management console views more usable

What This Document Contains

This book contains the following chapters:

- “Chapter 1. Introduction to the NetView Management Console” on page 3 describes the topology console and the topology server.
- “Chapter 2. Understanding Views” on page 5 describes some basic information about NMC views.
- “Chapter 3. Installing the NetView Management Console” on page 15 contains information about installing the NMC.
- “Chapter 4. Customizing the NetView Management Console” on page 17 contains information about customizing the NMC.
- “Chapter 5. Creating a Demo” on page 35 describes how to create NMC demos.

- "Chapter 6. Topology Console Java Applications and Plug-ins" on page 51 describes how to use Java™ applications and NetView management console plug-ins.
- "Chapter 7. Configuring Property Files for Locally Launched Applications" on page 61 describes the NMC web launch Java application.
- "Chapter 8. Operating the NetView Management Console" on page 71 contains information about operating NMC, including starting and stopping the topology server and topology console; as well as detailed information about the NMC window.
- "Chapter 9. Using the NMC Command Profile Editor" on page 87 describes how to control the content, order, and capabilities of the command menus, including adding commands and command sets to menus and defining command profiles for an individual operator or group of operators.
- "Chapter 10. Using the Topology Server Command Exits" on page 99 describes how to write command exits to use with NMC.
- "Chapter 11. Converting NGMF Command Sets" on page 131 describes how to convert a command set used with the NetView Graphic Monitor Facility (NGMF) to a command profile editor response file used with NMC.
- "Appendix A. Topology Server Commands" on page 141 contains a list of topology server commands.
- "Appendix B. Topology Console Commands" on page 161 contains a list of topology console commands.
- "Appendix C. Sending Commands to Multiple NetView for OS/390 Domains" on page 167 describes how to run a command against one or more NetView domains.
- "Appendix D. Automatic File Download at Console Log On" on page 169 describes how files are downloaded from the server to the console when the console signs on to the server.

Note: The windows shown in this book are examples; they might not exactly match the windows you will see while using NMC.

Conventions Used in This Document

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

Bold	Commands, keywords, flags, and other information that you must use literally appear like **this**, in **bold**.
Italics	Variables and new terms appear like *this*, in *italics*. Words and phrases that are emphasized also appear like *this*, in *italics*.
`Monospace`	Code examples, output, and system messages appear like `this`, in a `monospace` font.
ALL CAPS	Tivoli NetView for OS/390 commands are in ALL CAPITAL letters.

Platform-specific Information

For more information about the hardware and software requirements for NetView components, refer to the *Tivoli NetView for OS/390 Licensed Program Specification.*

Terminology

For a list of Tivoli NetView for OS/390 terms and definitions, refer to http://www.networking.ibm.com/nsg/nsgmain.htm.

For brevity and readability, the following terms are used in this document:

NetView

- Tivoli NetView for OS/390 Version 1 Release 4
- Tivoli NetView for OS/390 Version 1 Release 3
- TME® 10 NetView for OS/390 Version 1 Release 2
- TME 10 NetView for OS/390 Version 1 Release 1
- IBM NetView for MVS Version 3
- IBM NetView for MVS Version 2 Release 4
- IBM NetView Version 2 Release 3

MVS MVS/ESA™, OS/390, or z/OS operating systems.

Tivoli Enterprise™ software
Tivoli software that manages large business networks.

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. You may have used TME 10 environment in the past.

TME 10
In most product names, TME 10 has been changed to Tivoli.

V and R
Specifies the version and release.

VTAM® and TCP/IP
VTAM and TCP/IP for OS/390 are included in the IBM Communications Server for OS/390 element of the OS/390 operating system. Refer to http://www.software.ibm.com/enetwork/commserver/about/csos390.html.

Table 2 defines the Intel and UNIX® terms as they are used with the topology console and topology server. For example, on the topology server, the term Intel refers to Windows NT®, Windows® 2000, and OS/2® platforms.

Table 2. Topology Server/Topology Console Platforms

Term	Topology Console	Topology Server
Intel	• Windows 95 • Windows 98 • Windows NT • Windows 2000 • OS/2	• Windows NT • Windows 2000 • OS/2
UNIX	• AIX® • Solaris • HP/UX	• AIX

Unless otherwise indicated, references to programs indicate the latest version and release of the programs. If only a version is indicated, the reference is to all releases within that version.

When a reference is made about using a personal computer or workstation, any programmable workstation can be used.

Reading Syntax Diagrams

Syntax diagrams start with double arrowheads on the left (►►) and move along the main line until they end with two arrowheads facing each other (►◄).

As shown in the following table, syntax diagrams use ***position*** to indicate the required, optional, and default values for keywords, variables, and operands.

Table 3. How the Position of Syntax Diagram Elements Is Used

Element Position	Meaning
On the command line	Required
Above the command line	Default
Below the command line	Optional

Required Syntax

The command name, required keywords, variables, and operands are always on the main syntax line. Figure 1 specifies that the *resname* variable must be used for the CCPLOADF command.

CCPLOADF

Figure 1. Required Syntax Elements

Keywords and operands are written in uppercase letters. Lowercase letters indicate variables such as values or names that you supply. In Figure 2, MEMBER is an operand and *membername* is a variable that defines the name of the data set member for that operand.

TRANSMSG

Figure 2. Syntax for Variables

Optional Keywords and Variables

Optional keywords, variables, and operands are below the main syntax line. Figure 3 on page xiii specifies that the ID operand can be used for the DISPREG command, but is not required.

DISPREG

Figure 3. Optional Syntax Elements

Default Values

Default values are above the main syntax line. If the default is a keyword, it appears only above the main line. You can specify this keyword or allow it to default.

If an operand has a default value, the operand appears both above and below the main line. A value below the main line indicates that if you choose to specify the operand, you must also specify either the default value or another value shown. If you do not specify an operand, the default value above the main line is used.

Figure 4 shows the default keyword STEP above the main line and the rest of the optional keywords below the main line. It also shows the default values for operands `MODNAME=*` and `OPTION=*` above and below the main line.

RID

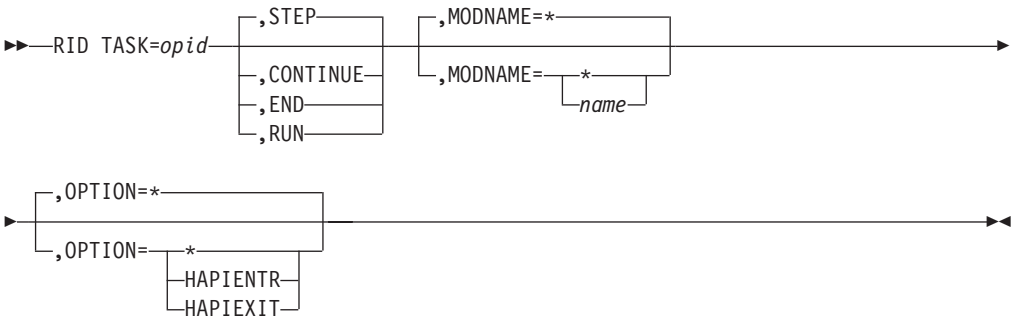


Figure 4. Sample of Defaults Syntax

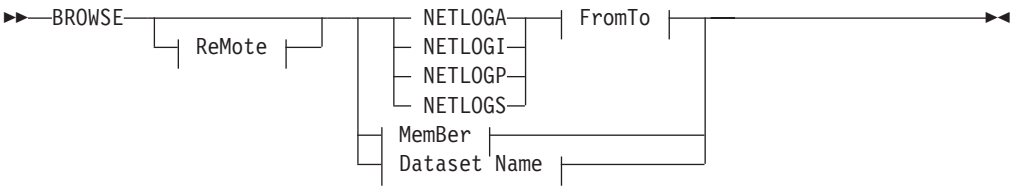
Long Syntax Diagrams

When more than one line is needed for a syntax diagram, the continued lines end with a single arrowhead (►). The following lines begin with a single arrowhead (►), as shown in Figure 4.

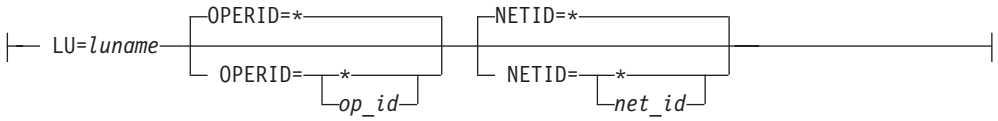
Syntax Fragments

Commands that contain lengthy groups or a section that is used more than once in a command are shown as separate fragments following the main diagram. The fragment name is shown in mixed case. See Figure 5 on page xiv for a syntax with the fragments `ReMote` and `FromTo`.

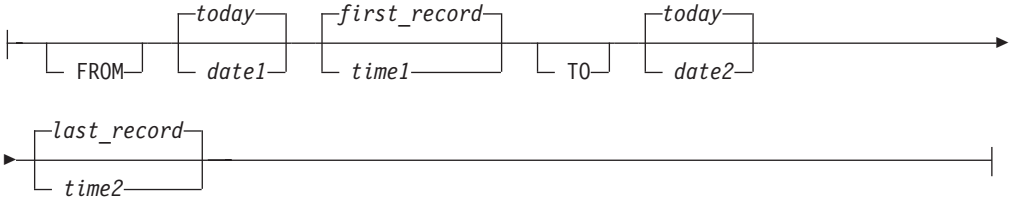
BROWSE



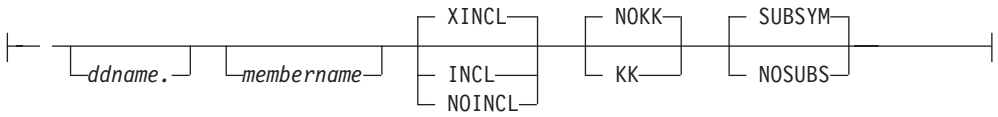
ReMote:



FromTo:



MemBer:



Dataset Name:

Figure 5. Sample Syntax Diagram with Fragments

Commas and Parentheses

Required commas and parentheses are included in the syntax diagram. When an operand has more than one value, the values are typically enclosed in parentheses and separated by commas. In Figure 6 on page xv, the OP operand, for example, contains commas to indicate that you can specify multiple values for the *testop* variable.

CSCF

PurgeBefore

Pu

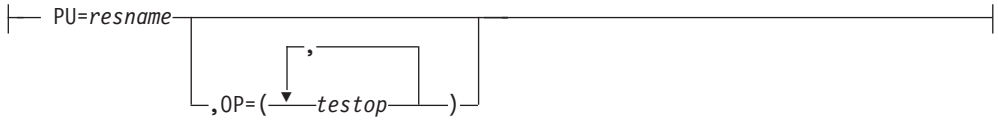


Figure 6. Sample Syntax Diagram with Commas

If a command requires positional commas to separate keywords and variables, the commas are shown before the keyword or variable, as in Figure 4 on page xiii.

For example, to specify the BOSESS command with the *sessid* variable, enter:

```
NCCF BOSESS applid,,sessid
```

You do not need to specify the trailing positional commas. Positional and non-positional trailing commas either are ignored or cause the command to be rejected. Restrictions for each command state whether trailing commas cause the command to be rejected.

Highlighting, Brackets, and Braces

Syntax diagrams do not rely on highlighting, underscoring, brackets, or braces; variables are shown italicized in hardcopy or in a differentiating color for NetView help and BookManager® online books.

In parameter descriptions, the appearance of syntax elements in a diagram immediately tells you the type of element. See Table 4 for the appearance of syntax elements.

Table 4. Syntax Elements Examples

This element...	Looks like this...
Keyword	CCPLOADF
Variable	*resname*
Operand	MEMBER=*membername*
Default	*today* or INCL

Abbreviations

Command and keyword abbreviations are described in synonym tables after each command description.

Accessing Publications Online

The Tivoli Customer Support Web site (http://www.tivoli.com/support/) offers a guide to support services (the *Customer Support Handbook*); frequently asked questions (FAQs); and technical information, including release notes, user's guides, redbooks, and white papers. You can access Tivoli publications online at http://www.tivoli.com/support/documents/. The documentation for some products is available in PDF and HTML formats. Translated documents are also available for some products.

To access most of the documentation, you need an ID and a password. To obtain an ID for use on the support Web site, go to http://www.tivoli.com/support/getting/.

Resellers should refer to http://www.tivoli.com/support/smb/index.html for more information about obtaining Tivoli technical documentation and support.

Business Partners should refer to "Ordering Publications" for more information about obtaining Tivoli technical documentation.

Note: Additional support is also available on the NETVIEW CFORUM (Customer Forum) through the IBMLink™ system. This forum is monitored by NetView developers who answer questions and provide guidance. When a problem with the code is found, you are asked to open an official problem management record (PMR) to get resolution.

Ordering Publications

Order Tivoli publications online at http://www.tivoli.com/support/Prodman/html/pub_order.html or by calling one of the following telephone numbers:

- U.S. customers: (800) 879-2755
- Canadian customers: (800) 426-4968

Providing Feedback about Publications

We are very interested in hearing about your experience with Tivoli products and documentation, and we welcome your suggestions for improvements. If you have comments or suggestions about our products and documentation, contact us in one of the following ways:

- Send e-mail to pubs@tivoli.com.
- Fill out our customer feedback survey at http://www.tivoli.com/support/survey/.

Contacting Customer Support

The *Tivoli Customer Support Handbook* at http://www.tivoli.com/support/handbook/ provides information about all aspects of Tivoli Customer Support, including the following:

- Registration and eligibility
- How to contact support, depending on the severity of your problem
- Telephone numbers and e-mail addresses, depending on the country you are in
- What information you should gather before contacting support

Part 1. Overview

Chapter 1. Introduction to the NetView Management Console

The NetView management console (NMC) of Tivoli NetView for OS/390 V1R4 graphically displays the resources that represent a network, a portion of a network, or a group of networks at various levels of detail. These views show the network and systems resources that you are monitoring. When you monitor a network, resource status changes are reflected graphically in the views.

What Can You Do with NetView Management Console?

You can use the NetView management console to do the following:

- Monitor and control large portions of complex business systems.
- View the topology and connectivity of your network graphically.
- Monitor the overall state of a network or a portion of a network through aggregates, which represent the combined status of a group of related applications and resources.
- Navigate easily from an aggregate to a real resource that is failing.
- Mark resources for your own purposes; for example, to show that they are being serviced.
- Display a list of events received for a selected resource.
- Issue predefined commands from context menus, or issue your own commands.
- Stop and restart selected resources.
- Specify which resources are critical to your network so that operators are notified when they are inactive.
- Monitor and manage multiple NetView programs.

NetView management console provides the added capability to create demos from your live NetView management console views. These demos can be used to aid in a variety of activities including:

- Operator training within your corporate setting
- Showing your customers what you can offer them
- NetView management console advocacy

You can create these demos by capturing live NetView management console views and integrating them into the demo. This enables you to make your demos look and feel like your real NetView management console system. For complete information about creating demos, see “Chapter 5. Creating a Demo” on page 35.

How Does NetView Management Console Work?

NetView management console consists of a *server* and a Java-based *console*, which are generically referred to in this book and the online help as the *topology server* and *topology console*.

The topology console graphically displays systems and networking information provided by Tivoli NetView for OS/390. This information is displayed as Resource Object Data Manager (RODM) based views and is only available if there is a conversation set up between the topology server and the NetView host. See “Chapter 8. Operating the NetView Management Console” on page 71 for information on setting up this conversation using the NETCONV command.

Topology Server

The topology server interacts with Graphic Monitor Facility Host Subsystem (GMFHS) and RODM and provides information for display on the topology console.

The topology server furnishes the topology console with a set of tasks that are applicable against a resource. These tasks appear in the context-sensitive menus on the topology console.

The topology server also stores files on behalf of the topology console, including icons, backgrounds, help files, log files, customized views, and settings.

Note that you must have a conversation between the topology server and the NetView host set up using the NETCONV command. See “Chapter 8. Operating the NetView Management Console” on page 71 for information on setting up communications with the NetView host.

Topology Console

The topology console graphically displays network information from the topology server. It displays systems and networking views. It uses color to indicate the status of each resource, and the status of the entire network. As the topology server receives configuration and status updates about the network, it updates the topology console. This enables the topology console to always display the real-time configuration of the network.

The topology console is a Java-based, platform-independent application.

Real and Aggregate Resources

The topology console can display both real and aggregate resources. A real resource is a single component or link (connection) in a network. An aggregate resource represents a collection of real or aggregate resources. When displayed on the topology console, an aggregate resource has a plus (+) in the lower-right corner, and an aggregate link has a plus (+) in the center of the link. The status of an aggregate resource is a reflection of the status of its underlying real resources. When you monitor an aggregate resource, you are monitoring the overall status of a portion of the network.

You can define real resources as critical and noncritical using aggregation priority. Critical resources are resources that are considered important to the operation of the network and are assigned a high aggregation priority (1 or greater). If the status of a critical resource changes to unsatisfactory, for example, the status of the aggregate resource would also change to degraded. Noncritical resources have low aggregation priorities (0).

You can set the aggregation priority on a resource from the Resource Properties notebook for a selected resource.

When a real resource changes status, the status of an aggregate of which the resource is a part is determined as follows:

- The status of the parent aggregate of the resource is determined by the statuses of the real resource and its siblings.
- The status of the grandparent aggregate of the resource is determined by the statuses of the real resources under the parent.

Chapter 2. Understanding Views

The NetView management console graphically displays systems and networking information provided by the NetView host. This information is displayed as Resource Object Data Manager (RODM) based views and is only available if there is a conversation set up between the topology server and the NetView host.

RODM-Based Views

RODM-based views are predefined in RODM or are dynamically built based on definitions in RODM. The Graphic Monitor Facility Host Subsystem (GMFHS) must be available to display RODM-based views and can include the following:

- Network views
- Exception views
- Configuration views
- More detail views
- Locate failing resource views

RODM-based views contain resources which are defined by the SNA topology manager (SNATM), MultiSystem Manager, user applications, loader files, and so on.

Note: Objects in the RODM GMFHS_Shadow_Objects_Class may be displayed in a view, (for example, a network view) but do not have any status. Commands against these resources are not supported.

Network Views

Network views and the resources displayed in them are defined in RODM. Figure 7 shows a network view.

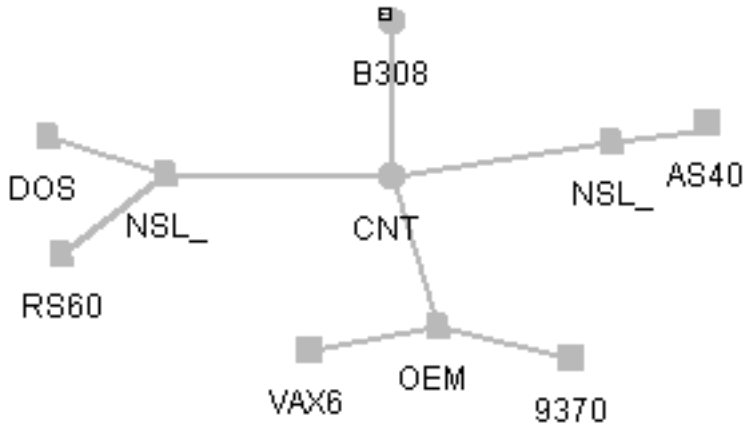


Figure 7. Network View

Exception Views

An exception view is a view that typically shows only resources that are not functioning properly, as defined by the exception criteria you defined in RODM. Figure 8 on page 6 shows an exception view.

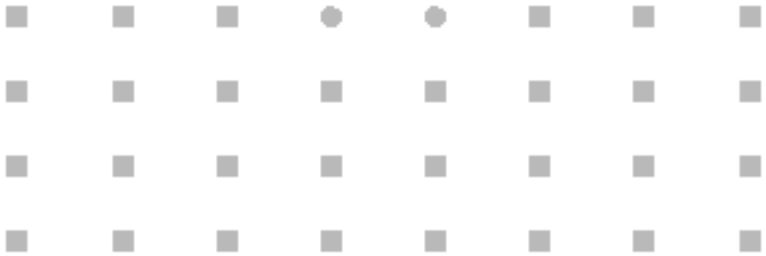


Figure 8. Exception View

Configuration Views

You can request these types of configuration views: parents, children, peers, logical, physical, logical and physical, and backbone. All relationships must have been previously defined in RODM.

Configuration Parents View

Figure 9 displays the configuration of a resource (not the entire connectivity) to its owning node.

Figure 9. Configuration Parents View

Configuration Children View

Figure 10 on page 7 shows the selected resource and all of its children.

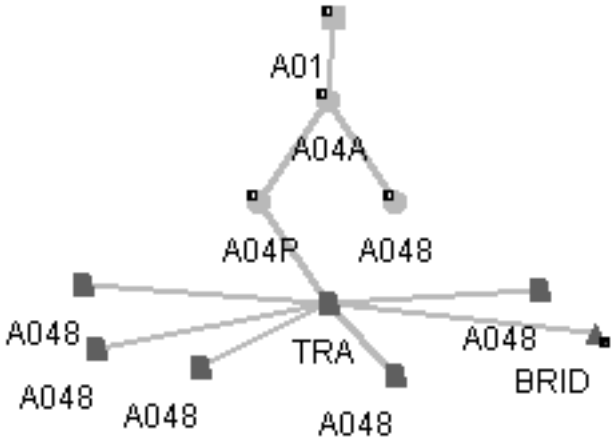


Figure 10. Configuration Children View

Configuration Peers View

Figure 11 shows a view containing resources in the network that are arranged in a configuration based on a peer relationship between resources.

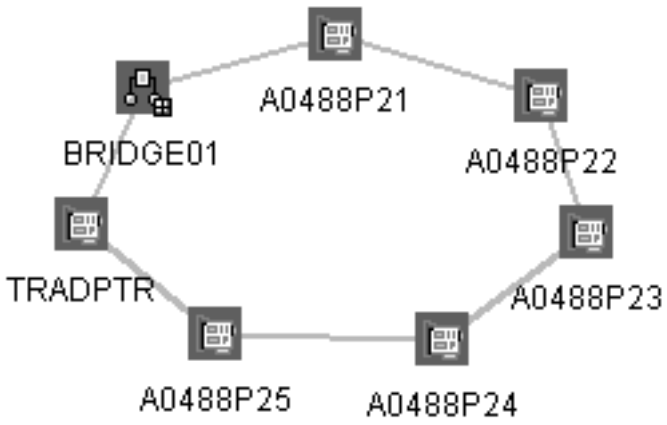


Figure 11. Configuration Peers View

Configuration Logical View

Figure 12 on page 8 shows a view containing resources in the network that are arranged in a configuration based on a logical relationship between resources.

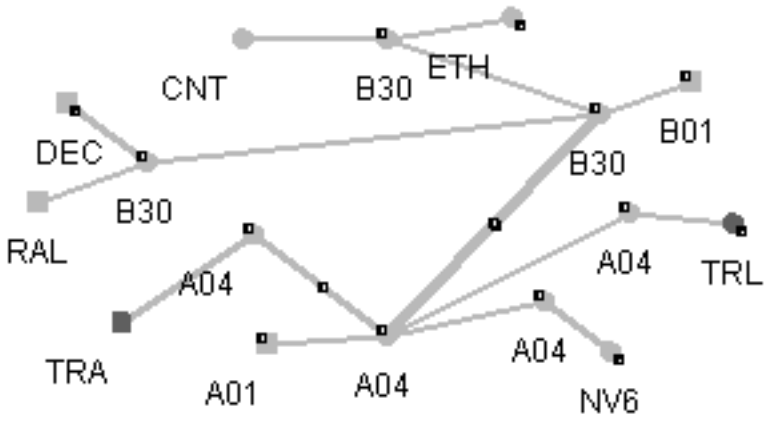


Figure 12. Configuration Logical View

Configuration Physical View

Figure 13 shows a view containing resources in the network that are arranged in a configuration based on a physical relationship between resources.

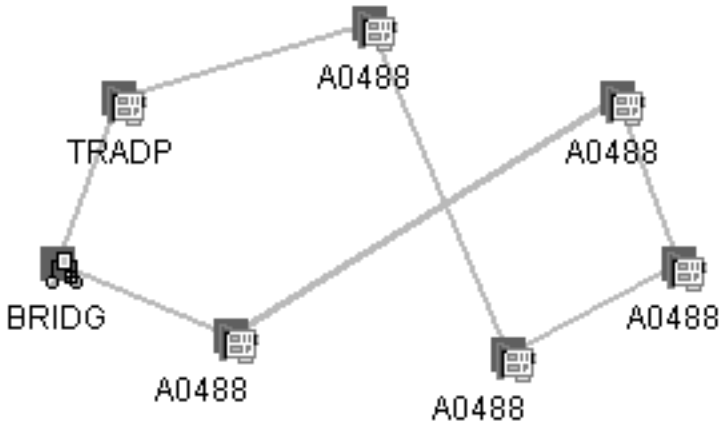


Figure 13. Configuration Physical View

Configuration Logical and Physical

Figure 14 on page 9 shows a view containing resources in the network that are arranged in a configuration based on a logical and physical relationship between resources.

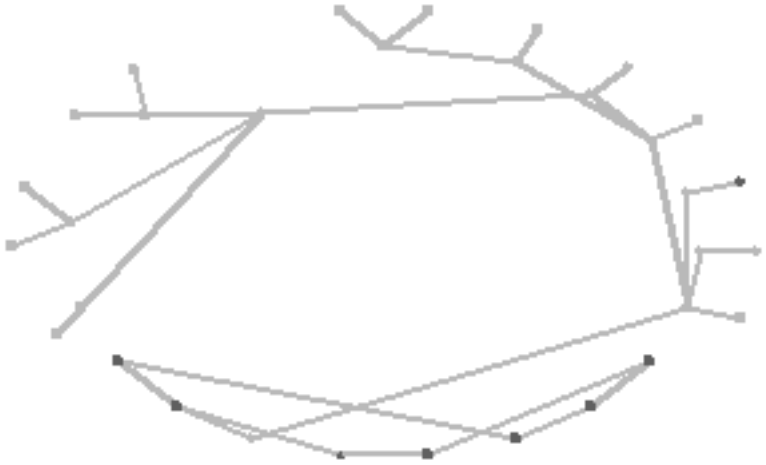


Figure 14. Configuration Logical and Physical View

Configuration Backbone View

Figure 15 shows a view containing resources in the network that are arranged in a configuration based on a subarea backbone relationship.

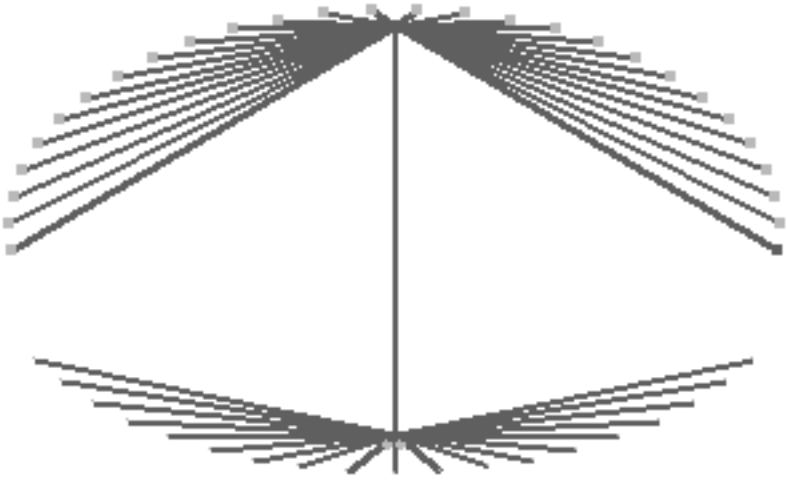


Figure 15. Configuration Backbone View

More Detail Views

When you request more detail about a selected resource, a view is displayed showing lower-level resources related to the selected resource. The More Detail function lets you navigate from high-level views to lower-level views. Figure 16 on page 10 shows the contents of a resource, or *more details* about the resource.

Figure 16. More Detail Views

Locate Failing Resources

Figure 17 shows a view which was created by selecting **Locate Failing Resources** on the context menu of an aggregate resource. This view displays all child real resources currently in an exception status.

Figure 17. Locate Failing Resource View

Customized Views

A customized view is a view that has been opened, changed, and saved using the Save View Customization function. You can use this function to save changes to views that are created dynamically in response to certain requests and to predefined RODM-based network views. If you are signed on as administrator, this function is available for the following types of views:

- Network views (predefined)
- Configuration views (both predefined and dynamically-built)
- More detail views (both predefined and dynamically-built)
- Locate failing resource views (dynamically-built)

If you are signed on as administrator, and there are dynamic views that have been customized, the Customized Dynamic Views node appears in the business tree.

Double-clicking a customized dynamic view in the business tree opens a snapshot of the view, but this snapshot is not an active view with real status and the latest topology changes. This enables you to see how dynamic views have been customized and to change the customization.

Note: Customized network views are shown in the business tree under Network Views.

Views Containing Resources for Which You Are Not Authorized

If you have defined span of control, some views can contain resources that you are not authorized to display because of your span authorization. When this occurs, the view is affected in one of the following ways, depending on your NetView customization:

- The unauthorized resources are not visible.
- The unauthorized resources are displayed as null nodes or null links, or both.

For more information about span of control, refer to the *Tivoli NetView for OS/390 Resource Object Data Manager and GMFHS Programmer's Guide*.

Views Containing Scheduled Resources

If you have defined NMCSTATUS policy definitions, a view can contain resources that are suspended from aggregation or are no longer receiving system status updates at the NetView management console. For more information about `NMCSTATUS` policy definitions, see the *Tivoli NetView for OS/390 Administration Reference*.

A resource that is suspended from aggregation because of a NMCSTATUS policy definition has a textual note attached to the suspend flag indicating why the resource was suspended. The note is displayed when either a Resource Properties or List Suspended Resources request is made.

A resource that is no longer receiving system status updates at the NMC console has a scheduled system status. The resource continues to receive system status updates in RODM but they are not sent to the NMC console while the resource is scheduled. As with other system statuses, you can customize the color of the scheduled system status on the Console Properties window.

Displaying Views in a Web Browser

You can configure the NMC console to function as a Web server. This enables the console to capture Topographic or Details NetView management console views and convert them into HTML and GIF files, which you can view in any Web browser.

Note: You can only display views that are open on the console machine.

If you want information about...	Refer to...
Displaying views in a web browser	"Configuring a Web Browser to Display Views" on page 19.
Using the NetView Resource Manager	"Customizing Web Server Enablement" on page 27.

Part 2. Installing and Customizing the NetView Management Console

Chapter 3. Installing the NetView Management Console

Installation instructions are provided in the installation README files. This chapter provides information on selecting the correct README file based on whether you are installing the topology server or topology console as well as information about configuring some portions of the NetView management console. For additional installation information on the various features of NMC, see the “Installing and Configuring the NetView Management Console for Graphics” chapter of the *Tivoli NetView for OS/390 Installation: Configuring Graphical Components* book.

The README files contain installation instructions, hardware and software requirements, and late-breaking news.

The README files are available on the Tivoli NetView for OS/390:

- CD-ROM in the README directory under the ENU or JPN subdirectories, where ENU represents English and JPN represents Japanese.
- Web page under the supported code section; and the URL for that is http://www.tivoli.com/nv390_supported.

Installing the Topology Console

For complete installation instructions for the topology console, see the EGVREAD1 README file (English) or the EGVREAD3 README file (Japanese).

Installing the Topology Server

For complete installation instructions for the topology server, see the EGVREAD2 README file (English) or the EGVREAD4 README file (Japanese).

Defining the NetView for OS/390 User ID and Password on the Topology Server

To use the **hostcmd** command to send commands from the topology server (using a command prompt on the topology server workstation) to the OS/390 environment, specify a NetView for OS/390 operator ID and password. There are three ways to define a NetView for OS/390 operator ID and password so that you are not prompted for them when using the **hostcmd** command.

1. Enter the **tserver hostcmd** command with the **-u** and **-p** options to set the user ID and password. This command also encrypts the password and stores it on disk.
2. Enter the **tserver hostcmdoper** command to set the user ID and password. This command encrypts the password and stores it on disk.
3. Customize the `ihsshstc.cfg` file as detailed in the following steps:
   a. Open a workstation command window.
   b. Change to one of the following directories:
      - For Intel: `%BINDIR%\TDS\server\config`
      - For UNIX: $BINDIR/TDS/server/config
   c. Edit the ihsshstc.cfg file to specify the operator ID and password of the NetView operator for which the commands will be executed.
      1) Enter the NetView operator ID in the OPER_ID parameter.
      2) Enter the password for that operator ID in the OPER_PW parameter.

Note: The userid and password are accessed in this order.

Chapter 4. Customizing the NetView Management Console

This chapter describes basic customization of the NetView management console as well as advanced customization that can be performed to change the look, feel and function of the NetView management console.

Customizing Topology Console Features

You can customize features of the NetView management console topology console by adding or changing any of the following:

- Topology console icons
- Topology console backgrounds
- The number of history items for various topology console functions
- Topology console help

Customization is performed at the topology server so that it automatically deploys to each topology console that subsequently signs on.

Adding and Customizing Topology Console Icons

To add a new icon to the topology console, create an icon in one of the following graphical interchange formats (GIFs):

32x32 pixels
This size is required.

The file must be named 32_xxxx.gif (using only lowercase letters).

24x24 pixels
This size is optional and is used for the medium view sizes. If this size is not provided, the 32_xxxx.gif version will be automatically scaled.

The file must be named 24_xxxx.gif (using only lowercase letters).

16x16 pixels
This size is optional and is used for the smaller view sizes. If this size is not provided, the 32_xxxx.gif version will automatically be scaled.

The file must be named 16_xxxx.gif (using only lowercase letters).

Note: Animated GIFs are not supported.
Place the GIF files in one of the following directories:

- For Intel: `%BINDIR%\TDS\server\db\current\icons`
- For UNIX: `$BINDIR/TDS/server/db/current/icons`

You can also change existing icons in these directories using any tool that enables GIF file manipulation.

One example of a customized icon is the company icon. To customize this icon, use the Console Properties notebook. For details, see the icon customization procedures in the NetView management console online help.

Adding and Customizing Topology Console Backgrounds

To add a new background image to the topology console, create an image file following these guidelines:

- The name of the file must contain only lowercase letters.

- For a GIF image, use **gif** for the extension (or file type).
- For a JPEG image, use **jpg** for the extension (or file type).
- The recommended size is 300x500 pixels.

Store the image file in the appropriate topology server directory:

- For Intel: `%BINDIR%\TDS\server\db\current\backgrounds`
- For UNIX: `$BINDIR/TDS/server/db/current/backgrounds`

You can also change the existing backgrounds in these directories using any tool that enables GIF or JPG manipulation.

Customized backgrounds are associated with a view. See the NetView management console online help for specific steps on how to customize a view background.

Note: Removing unused backgrounds reduces the install download time for the topology console.

Displaying Customized Help

You can create context menu help that displays a Hypertext Markup Language (HTML) document at the topology console using the following instructions.

1. Create your document file using basic HTML tags.

   Note: Use only lowercase characters in the file name and extension.
2. Place the help file into the appropriate directory on the topology server workstation:
   - For Intel: `%BINDIR%\TDS\server\db\current\help`
   - For UNIX: `$BINDIR/TDS/server/db/current/help`

   Note: You can also change existing help files which are located in these directories.
3. Add the new context command to the appropriate command profile editor, using the GUI as follows:
   a. From the **Command** notebook, create a new command with the following values.

      Command string field:

      `com.tivoli.ihs.client.action.IhsShowDocument <document name>`

      Command exit field:

      `IHSXTJAM`

   b. Click **Add**.

   The business system help command shipped with the product provides an existing command that can be studied or copied.

   Note: You can also use the command profile editor batch utility to add help.
4. Add the command to the default profile as follows:
   a. From the **Profiles** notebook, select the default profile.
   b. Click **Profile** → **Add existing**.
   c. Click **Command** and select the newly created command to add to the profile.

To see an example of defining a command and adding it to a command profile for UNIX platforms, use the command profile editor batch utility and refer to the sample `ihsscpe.xxx.rsp` (where *xxx* is a country code indicator, such as en_US) in the sample topology server directory.

Configuring a Web Browser to Display Views

Designating a Console as a Web Server

When you first install the NetView management console, the Web server function is not enabled. Use the Web Server page in the Console Properties notebook to designate the console as a Web server. Click the **Help** button on the Web Server page for detailed descriptions of the page settings.

Designating Multiple Consoles as Web Servers

You can designate multiple consoles on the same machine as web servers, so that each console can set up its own set of views. Supply the data on the Web Server page in the Console Properties notebook to designate each console as a Web server, but assign a unique port number to each console on the same machine. Web servers on different machines can have the same port number.

Using the Web Browser

After designating the console machine as a Web server, enter the fully qualified host name or IP address of the console machine as a URL in the Web browser. You do not need to add prefixes (http, www, etc.), although you might need to fully qualify the host name. If you entered a port number other than the default (80) in the **Port Number** field on the Web Server page of the Console Properties notebook, follow the host name with a colon and the port number in the browser as well. If you keep the default port number, the URL should appear as follows:

```
Clientname
```

If you enter a port number other than the default, the URL should appear as follows:

```
Clientname:Port number
```

You will not see any views in the Web browser until you add the view to the set of available views on the console machine. Once specified, the view appears in the Web browser after the next Web browser refresh interval. When a view appears, the following information appears above the view:

- The number of available views
- The name of the current view
- The time and date that the view was captured

The refresh interval appears below the view.

Web browser views are not dynamic: they are *snapshots* in time of a view that is opened on the console machine. If the status of a view changes, the change appears in the browser after the next browser refresh interval. As views are captured for the Web server by the console machine, they are added to a list of views. To capture the view, it must be visible on the console.

If your console machine has been configured to use the Cycle Views function, the views on the console are automatically displayed for a certain time interval, enabling the views to be captured for the Web server automatically. To use the Cycle Views function to select the views and define the time interval to display them, from the Windows menu, select **Cycle Views**.

Topology views in the Web browser are the same size as the views captured on the console machine. To change the size of a topology view in the Web browser, re-size the view on the console machine and select **Add View to Web Server** or **Update View to Web Server** from the pop-up menu. The re-sized view will appear in the Web browser at the next refresh, or you can manually reload the view in the Web browser by clicking the **Reload** button.

A list of available views is provided in the browser window. Select a view, and click **Open**. As additional views are added to the set of available views on the console, they are added to the view list. Both Topology and Details views can be displayed as they are displayed on the console. Icons in the Details view will not contain flags, background color, or an aggregate symbol.

You can manually refresh a view by right-clicking the view background on the console machine. From the pop-up menu, select one of the following:

- **Add View to Web Server**
- **Update View on Web Server**
- **Refresh Now** (if the view has been added)

The view will be captured and sent to the browser. Click **Reload** from the Web browser or wait for the next refresh interval to see the refreshed view. If you did not select **Make Views Available When They Are Opened** from the **Web Server** page, you must manually add views to the Web browser. To do this, right-click the view background and select **Add View To Web Server** from the pop-up menu. The view will appear in the list of views available to the Web browser at the browser's next refresh interval.

Logging Web Server Messages

You can indicate on the Web Server page in the Console Properties notebook to create a record of each view that is opened on the web browser and send the record to the console log. This is not recommended unless you need to closely track your views, since it creates a record every time a view is refreshed from the attached browsers.

Advanced Topology Console Customization

There are several advanced customization tasks for the topology console. Most of these tasks can be performed without using the topology console interface.

Customizing Your Online Help Facility

As an alternative to using the built-in NetView management console help facility, you can specify your own Web browser to display HTML help pages. Select the **General** tab on the **Console Properties** notebook. In the group box labeled **Configure help facility**, select one of the following options:

- Use the built-in help facility.
- Use my default Web browser.
- Let me specify my own browser.

To specify your own Web browser, use lower case only, and specify the full drive and path name for the location of the Web browser executable file. Select **Browse** to navigate to the desired directory and locate the executable file. If you specify $URL anywhere in the specified path, the Help screen URL is substituted for the $URL automatically. If you do not specify a $URL, the Help screen URL is appended to the end of the given command path. If you use a blank in your path name, enclose the entire command in quotation marks.

When online help is selected from the menu, the corresponding HTML help pages are sent to the browser of your choice. An HTML message displays in either case (if there are any problems launching the Web browser or after successfully launching the Web browser).

Note: If you request help at the **Sign On** dialog, the help is displayed in the NetView management console help facility, regardless of your choice of help facility in this Console Properties settings page. At the time you sign on, the values specified on the console settings page are not yet available.

Enabling User Flags

Thirty-two flags are shipped with the NetView management console; eight of these can be customized. When they are shipped from the factory, the values for all of them are disabled. Therefore, if one of these flags is set for a resource, it will not display on any NetView management console user interface component. To display these flags, they must first be enabled. After a customized flag has been enabled, it displays on the appropriate NetView management console user interface component (such as, in the Resource Properties window, or the Filter Bar).

To enable any of the flags that can be customized, update the default operational scheme by performing the following steps from the server workstation:

1. Make a backup copy of the NetView management console default operational scheme files with one of the following methods:
   - Use a packaging tool such as PKZip, WinZip, or tar.
   - Create a backup directory and copy the default scheme files to it.
2. Determine the hexadecimal value of the specific user flag that you want to enable by using the following steps:
   a. Open the defaultscheme.properties file in a text editor.
   b. Search for the string `f1` to locate the section of the file where the flags are defined.
   c. Scroll down until you find the comment for the flag you want to enable.

      For example:

      ```
      * User 1
      f25.value =0x00000080
      ```

      The corresponding value of the attribute is the hexadecimal value for the flag. In this example, `0x00000080` is the `User 1 flag` value.
3. In the defaultscheme.properties file, search for the string `FLAG values` to locate the section of the file where the flags are defined. The flags are defined in descending order.
4. Scroll down until you see the value for the user flag that you want to enable.
5. To enable this flag, change the value of the `defDefine` attribute from `false` to `true`.

   Depending on how you want this flag to work, you might need to define additional attributes. All flag attributes are documented by the `com.tivoli.ihs.client.view.IhsUserStatus` class as shown in the following example.

   ```
   #  "Flag" Definition Values:
   #
   #    fX.tag         Reference tag (required).
   #    fX.defDefine   Is this item defined? (optional, true).
   #                   Set to false to disable this item.
   #    fX.isDefault   Does this item contain default values for all other items?
   ```

```
#                   (optional, false).
#    fX.weight      Orders an item relative to other items by "weight"
#                   (optional, 100).
#    fX.value       Status value (required, only 1 bit can be on).
#    fX.defFilter   Currently not used.
#    fX.defDisplay  Currently not used.
#
#    fX.onView      Display sub-icon on topology view? (optional, false)
#    fX.color       Color of sub-icon on topology view.  Flag with highest
#                   weight is used. (optional, gray)
#    fX.reqAdmin    Administrator required to set/clear (optional, false).
#    fX.canSet      Is this flag allowed to be set (turned on)? (optional, true)
#    fX.canClear    Is this flag allowed to be cleared (turned off)? (optional, true)
#    fX.forAgg      Applicable for an aggregate resource? (optional, false)
#    fX.forReal     Applicable for a real resource? (optional, true)
#    fX.relatedTo   Mask of "related" flags (optional, none).
#
```

For more information on self-documenting data classes for the NMC console, see “Running a Console Class” on page 27.

6. Save your changes.
7. To define the wording for this user flag, open the defaultschemetext.properties file in a text editor.
8. Search for the `f.`*`xxxxxxx`* string where *xxxxxxx* is the hexadecimal value of the specific user flag you are enabling.
9. Change the value of the `f.`*`xxxxxxx`*`.label` attribute to include all text for this flag.
10. Change the value of the `f.`*`xxxxxxx`*`.abbrev` attribute to the abbreviation you are assigning to this flag.
11. Save the changes.

To verify that the flag you have enabled exists, perform the following steps:

1. Start an NMC console.
2. Open a view.
3. Select a resource and right-click to display its context menu.
4. Click **Resource Properties**.
5. Verify that the user flag is present on the **Resource** window.
6. Open the **Console Properties** notebook, select the **Status** page, and verify that the flag is present.

A Flag Enablement Example

The example in Table 5 shows the changes that were made to enable the `User 1 flag` and name it `Retired`.

Table 5. Changing the User 1 flag to retired.

File	Before	After
defaultscheme.properties	* User 1 f25.value =0x00000080 f25.weight =2000 f25.defDefine =false	* User 1 f25.value = 0x00000080 f25.weight = 2000 f25.defDefine = true
defaultschemetext.properties	f.00000080.label = User 1 f.00000080.abbrev = User1	f.00000080.label =Retired f.00000080.abbrev =Retd

Adding a Flag to the Context Menu

The following flags can be directly manipulated from resource specific context menus:

- Suspend, Manually Clear
- Suspend, Automatically Clear
- Clear Suspended
- Clear Child Suspended

You can enable context menu items for additional flags. From the server workstation, add a new flag context menu item to the NetView management console default operational scheme. The properties files, in which the NetView management console operational scheme is defined, are described briefly in Table 6.

Table 6. Description of NetView management console properties files

Scheme File Name	Description
defaultscheme.properties	Provides most of the operational definitions (except required NLS enabled text). 1. COLOR Definitions (Not currently used) 2. FLAG Definitions 3. FLAG (User status) menu items 4. STATUS SCHEME values 5. STATUS values 6. STATUS MAPPING values (Not currently used) 7. MODE values (NetView management console only supports Control mode) 8. LAYER values 9. Miscellaneous Control Values
defaultschemetext.properties	Provides the English set of NLS enabled text that is required by the `defaultscheme.properties` file. *(Status text and colors not currently used.)*
defaultschemetext_ja.properties	Provides the Japanese set of NLS enabled text that is required by `defaultscheme.properties` file. *(Colors not currently used.)*

The disk location of the scheme properties files is determined by the NetView management console operational mode (as shown in Table 7).

Table 7. Where to edit scheme files, based on mode.

NetView management console Operational Mode	Location of Scheme Files
Signed on to a server	<installRoot>\bin\<*interp*>\TDS\Server\db\current\settings
Demo	<installRoot>\bin\generic_unix\TDS\client\settings

To add a new flag context menu item to the NetView management console default operational scheme, perform the following steps:

1. Make a backup copy of the NetView management console default operational scheme with one of the following methods:
   - Use a packaging tool such as PKZip, WinZip, or tar.

- Create a backup directory and copy the default scheme files to it.

2. Determine the hexadecimal value of the specific user flag that you want to enable by performing the following steps.
   a. Open the `defaultscheme.properties` file in a text editor.
   b. Search for the string `f1` to locate the section of the file where the flags are defined.
   c. Scroll down until you find the comment for the flag.

      For example:

      ```
      * Marked
      f1.value =0x80000000
      ```

      The corresponding value for the attribute is the hexadecimal value for the flag. In this example, `0x80000000` is the `Marked flag` value.
3. In the defaultscheme.properties file, search for the string `User status menu items` to locate the section of the file where the context menu items for the flag are defined.
4. The menu items are defined in the order they will appear in a context menu, so scroll down to the bottom of the list of menu items.
5. Create a new set of menu item attributes with a `menu`*x* prefix (where *x* is the next sequence number). For example:

   ```
   menux.tag   = <unique identifier for this menu item>
   menux.value = <hex value of the flag that you want to enable>
   menux.setTo = <true=set flag -or- false=Clear flag>
   ```
6. Depending on how you want this context menu item to work, you might also have to define additional attributes.

   The `com.tivoli.ihs.client.view.IhsFlagMenuScheme` class documents all menu item attributes, as shown in the following example.

   ```
   #  "User Status Menu Item" Definition Values:
   #
   #  defaultscheme.properties:
   #    menuX.tag         Reference tag (required).
   #    menuX.defDefine   Is this item defined? (optional, true).
   #                      Set to false to disable this item.
   #    menuX.isDefault   Does this item contain default values for all other items?
   #                      (optional, false).
   #    menuX.weight      Orders an item relative to other items by "weight"
   #                      (optional, 100).
   #    menuX.value       Associated flag value (required, only 1 bit can be on).
   #    menuX.setTo       When selected, set flag on (true) or off (false) (required).
   #    menuX.setRel      If this flag is set, should a related flag also be set?
   #                      (optional, false).
   #                      See flag definitions relatedTo setting for the related flag.
   #    menuX.setRelTo    If the related flag is to be set, set it on (true) or off (false)
   #                      (optional).
   #  defaultschemetext.properties:
   #    menuX.text        Menu text (required)
   #    menuX.help        Menu help file (required)
   #
   ```

   For more information on self-documenting data classes for the NMC console, see “Running a Console Class” on page 27.
7. Save the changes.
8. Open the `defaultschemetext.properties` file in a text editor and search for the string `User status menu items text` to locate the section of the file where the flag menu items are defined.
9. Scroll down to the end of the list of menu items which are currently defined.

10. Create a new attribute with prefix that is the same as the `menu`*`x`* value you used in the defaultscheme.properties file.

    For example:

    ```
    menux.text= <text displayed for this menu item>
    ```

11. Save the changes.

To verify these changes, perform the following steps:

1. Start an NMC console.
2. Verify that the new context menu is displayed properly:
   a. Open a view.
   b. Locate a resource for which your new menu item should be available.
   c. Right click on that resource and verify that your new menu item is present.
   d. Select the menu item and verify that the flag has been updated.
3. Verify that the definition of the default note has been updated:
   a. Open the **Console Properties** notebook and select the **Notes** tab.
   b. Verify that the new menu item is present.

For Resource Object Data Manager (RODM)-based resources, the NetView management console flags are the 4-byte `UserStatus` fields. See the *Tivoli NetView for OS/390 Data Model Reference* manual for details.

Adding Flag Examples

The example in Table 8 on page 26 shows the changes made to enable context menu items for `Marked`*`flag`* in the defaultscheme.properties file.

Table 8. Adding the Markedflag item to the defaultscheme.properties file.

Before	After
**********	**********
* User status menu items	* User status menu items
**********	**********
menu1.setRelTo = false	menu1.setRel = true
menu2.tag = SuspAut	menu1.setRelTo = false
menu2.value = 0*x*40000000	menu2.tag = SuspAut
menu2.setTo = true	menu2.value = 0*x*40000000
menu2.setRel = true	menu2.setTo = true
menu2.setRelTo = true	menu2.setRel = true
menu3.tag = ClearSusp	menu2.setRelTo = true
menu3.value = 0*x*40000000	menu3.tag = ClearSusp
menu3.setTo = false	menu3.value = 0*x*40000000
menu3.setRel = true	menu3.setTo = false
menu3.setRelTo = false	menu3.setRel = true
menu4.tag = ClearChildSusp	menu3.setRelTo = false
menu4.value = 0*x*00800000	menu4.tag = ClearChildSusp
menu4.setTo = false	menu4.value = 0*x*00800000
	menu4.setTo = false
	menu5.tag = SetMarked
	menu5.value = 0*x*80000000
	menu5.setTo = true
	menu6.tag = ClearMarked
	menu6.value = 0*x*80000000
	menu6.setTo = false

The example in Table 9 on page 27 shows the changes that were made to enable context menu items for the `Marked`*`flag`* in the defaultschemetext.properties file.

Table 9. Adding the Markedflag item to the defaultschemetext.properties file.

Before	After
******************************	********************************
* User status menu items text	* User status menu items text
******************************	********************************
menu1.text = Suspend, Manually Clear	menu1.text = Suspend, Manually Clear
menu1.help = ihs_mi_clermansusp_*xxx*.html	menu1.help = ihs_mi_clermansusp_*xxx*.html
menu2.text = Suspend, Automatically Clear	menu2.text = Suspend, Automatically Clear
menu2.help = ihs_mi_clerautosusp_*xxx*.html	menu2.help = ihs_mi_clerautosusp_*xxx*.html
menu3.text = Clear Suspended	menu3.text = Clear Suspended
menu3.help = ihs_mi_clersuspended_*xxx*.html	menu3.help = ihs_mi_clersuspended_*xxx*.html
menu4.text = Clear Child Suspended	menu4.text = Clear Child Suspended
menu4.help = ihs_mi_clerchldsusp_*xxx*.html	menu4.help = ihs_mi_clerchldsusp_*xxx*.html
	menu5.text = Set Marked
	menu6.text = Clear Marked

Running a Console Class

Many of the NMC console data classes are self-documenting. This means that if you run the data class, it tells you all of the attributes that you can define in a properties file to create an instance of the class. All of the NMC console classes are contained in the `\bin\generic_unix\tds\client\lib\ihseuc.jar` file. To obtain information for a specific data class, issue the following command from a console workstation:

```
java -classpath %classpath%;<installRoot>\bin\generic_unix\tds\client\lib\ihseuc.jar
<package qualified class name>
```

This command generates all of the attributes for a flag, for example:

```
java -classpath %classpath%;<installRoot>\bin\generic_unix\tds\client\lib\ihseuc.jar
com.tivoli.ihs.client.view.IhsUserStatus
```

Customizing Web Server Enablement

You can enable or disable the Web server function (as a tab on the Console Properties notebook). Table 10 shows the values for enabling or disabling the Web browser function; these values are set in the defaultscheme.properties file.

Table 10. Changing line thickness (in pixels).

`defaultscheme.properties` Attribute	Default Value	Other valid value
webServerPage.enable	true	false

To customize the Web Server enablement, the NetView management console operational scheme must be updated, as shown in Table 10. To customize Web server enablement, perform the following steps.

1. Make a backup copy of the default NetView management console operational scheme.
2. Open the `defaultscheme.properties` file in a text editor.
3. Search for the string `webServerPage`.
4. Change the value on this line to false if you want to disable the Web Server function.
5. Save the changes.

To verify this change, perform the following steps:

1. Start an NMC console
2. Open the Console Properties notebook.
3. Verify that the Web Server tab no longer appears on the Console Properties notebook page.

Note: Setting the attribute in the defaultscheme.properties file overrides the automatic start setting. For example, if you have previously set the Web Server to automatically start and then set `webServerPage.enable=false`, then the Web Server will not automatically start.

For more information on the NetView management console operational scheme, see Table 6 on page 23.

Customizing the View Bar Layout

You can adjust the number of rows or columns of view buttons displayed on the View Bar of the main NetView management console window. The minimum size of the view buttons can also be adjusted. Table 11 shows the attributes used to adjust these settings. These attributes are set in the defaultscheme.properties file.

Note: Either the number of rows *or* the number of columns can be customized, but not both. The value for one of these settings must remain 0. The 0 value indicates that no maximum value exists for the attribute.

Table 11. Changing View Bar attributes and values in the defaultscheme.properties file.

Attribute	Description	Default Value	Minimum Value	Maximum Value
`viewbar_max_rows = <int>`	Maximum number of rows in the view bar grid. New columns will be added to display additional views.	0		None
`viewbar_ max_columns = <int>`	Maximum number of columns in the view bar grid. New rows will be added to display additional views.	0		None
`viewbar_min_view_size = <int>`	Minimum size to make a view when sizing the views to fit on the view bar.	100	100	

To customize the grid so that the view buttons appear on the View Bar as you want them to appear, update the NetView management console operational scheme from the console or server workstation, depending on the mode in which you are running. See Table 7 on page 23 to determine if the updates should be performed from the server or console workstation. To customize the view bar layout, perform the following steps.

1. Make a backup copy of the default NetView management console operational scheme.
2. Open the defaultscheme.properties file in a text editor.
3. Search for the string `viewbar` and locate the attribute to be changed.
4. Change the value of the attribute.
5. Save the changes.

View Bar Row Example

The example in Table 12 shows the changes made to the maximum number of rows in the View Bar grid from 0 to 5.

Table 12. Changing the number of rows in a view bar in the defaultscheme.properties file.

Before	After
`viewbar_max_rows = 0`	`viewbar_max_rows = 5`

For more information on the NetView management console operational scheme, see Table 6 on page 23.

Customizing the Automatic Download of Files At Log On

When the console logs on to the server, the timestamp of the files (as stored on the console) and the timestamp of the files on the server are compared. If the console consistently logs on to the same server, these timestamps are the same and the files are not downloaded. See “Appendix D. Automatic File Download at Console Log On” on page 169 for more information. If the console logs on to servers on different platforms, however, then these timestamps will be slightly different. The defaultscheme.properties file enables you to set the tolerance level of the time difference between timestamps. The tolerance level is a value, in minutes, between the times on the timestamps. If it is set to 0 (zero), for example, then the files are automatically downloaded from the server unless the server and console timestamps match exactly. Table 13 shows the attribute and the range of valid values for this attribute.

Table 13. Changing the tolerance for timestamp differences in the defaultscheme.properties file

`defaultscheme.properties` attribute	**Default Value**	**Minimum Value**	**Maximum Value**
`file_download_timestamp_tolerance = int`	720 (12 hours)	0	43,200 (30 days)

A Timestamp Tolerance Example

The example in Table 14 on page 30 shows two scenarios in which the timestamp tolerance can be used and the associated values.

Table 14. Examples of timestamp tolerance settings in the defaultscheme.properties file

Scenario:	Value(s) to use:
The console is logging on to the same server and you want to automatically download any files that have changed each time you log on.	Set the value to 0 (zero) as follows: `file_download_timestamp_tolerance = 0`
You are logging on to multiple servers on various platforms and you do not want the files to download each time you log on	Do one of the following: • leave the default value of 12 hours:`file_download_timestamp_tolerance = 720` • set the value higher, three days in this example: `file_download_timestamp_tolerance = 4320` • set the value lower, six hours in this example: `file_download_timestamp_tolerance = 360`

Overriding the Default Date/Time Format

By default, the Java run time obtains display formats for the date and time from the operating system. On Windows NT, for example, the date and time display formats are defined by selecting **Control Panel → Regional Settings**. Unfortunately, certain changes, such as the time style, that are made to these values are not propagated to Java™2 Runtime Environment, Standard Edition, v13 (J2RE). If you change the `Time` style in **Regional Settings**, in Windows/NT, to display in 24 hour format, this change is not propagated to the J2RE. The topology console, therefore, appears to be out of sync with the rest of the local applications.

You can force the console to override the use of the operating system formats and display the dates and times you want. Table 15 shows the attributes (in the defaultschemetext.properties file) used to make this change.

Table 15. Changing the date/time format in the defaultschemetext.properties file.

Attributes	Description
`override.date = <format>`	format used when just formatting a date
`override.time = <format>`	format used when just formatting a time
`override.datetime = <format>`	format used when formatting a composite date and time

The example in Table 16 shows how a time stamp of 6:45:07 PM on March 30, 2001 displays for various format specifications.

Table 16. An example of time stamp format specifications.

Format Specification	Resulting Display Text
yyyy.MM.dd	2001.03.30
yyyy.MMM.dd	2001.Mar.30
yyyy.MMMM.dd	2001.March.30
hh:mm:ss a	06:45:07 PM
hh:mm:ssa z	06:45:07PM EST
HH:mm:ss z	18:45:07 EST

To customize the date/time format, update the default operational scheme from the console or the server workstation, depending on the mode in which you are running. See Table 7 on page 23 to determine if the updates should be performed from the server or console workstation. To customize the date/time format, perform the following steps.

1. Make a backup of the default NetView management console operational scheme.
2. Open the defaultschemetext.properties file in a text editor.
3. Search for the string `override.date`.
4. Uncomment the desired override item by removing the leading * from the beginning of the line.
5. Change the value on the `<value>` line to the format specification you want.

   The `<value>` field defines the format of the displayed item. It can contain both literal and substitution symbols.
6. Save the changes.

To verify the changes, perform the following steps:

1. Start an NMC console.
2. Open a view.
3. Verify the `override.datetime` attribute change by checking the view information area in the status bar.
4. Verify the `override.date` and `override.time` attribute changes by placing the view in *details* mode and checking the Time/Date column.

A Date and Time Override Example

The example in Table 17 shows the changes made to override all of the date/time display formats. These changes were made in the defaultschemetext.properties file.

Table 17. Overriding the date/time display formats in the defaultschemetext.properties file.

Before	After
**********	**********
* Date/Time Overrides Formats	* Date/Time Overrides Formats
**********	**********
* override.date =yyyy.MMM.dd	override.date =yyyy.MMM.dd
* override.time =HH:mm:ss z	override.time =HH:mm:ss z
* override.datetime=yyyy.MMM.dd @ HH:mm:ss z	override.datetime=yyyy.MMM.dd @ HH:mm:ss z

For more information on the NetView management console operational scheme, see Table 6 on page 23.

Formatting Capabilities: The Time Format Syntax

The example in Table 18 on page 32 is an excerpt from the javadoc of the `java.text.SimpleDateFormat` class. It provides details about all of the possible formatting capabilities. To specify the time format, use a time pattern string. In this pattern, all ASCII letters are reserved as pattern letters, which are defined in Table 18 on page 32.

Table 18. The time format syntax: formatting capabilities.

Symbol	Meaning	Presentation	Example
G	era designator	(Text)	AD
y	year	(Number)	2001
M	month in year	(Text & Number)	April & 01
d	day in month	(Number)	10
h	hour in am/pm (1-12)	(Number)	12
H	hour in day (0-23)	(Number)	0
m	minute in hour	(Number)	30
s	second in minute	(Number)	55
S	millisecond	(Number)	978
E	day in week	(Text)	Tuesday
D	day in year	(Number)	189
F	day of week in month	(Number)	2 (2nd Wed in July)
w	week in year	(Number)	27
W	week in month	(Number)	2
a	am/pm marker	(Text)	PM
k	hour in day (1-24)	(Number)	24
K	hour in am/pm (0-11)	(Number)	0
z	time zone	(Text)	Pacific Standard Time
’	escape for text	(Delimiter)	
’ ’	single quotation mark	(Literal)	’

The count of pattern letters determines the format, as follows.

- Text: If the presentation is in text and there are 4 or more pattern letters, then use the full form. If there are less than 4 pattern letters, then use the short or abbreviated form, if one exists.
- Numeric: If the presentation is numeric, then the field will contain the minimum number of digits. Shorter numbers are zero-padded to this amount.

  Note: The year is handled differently. If the count of contents of the y field is 2, then the year will be truncated to 2 digits.
- Text and numeric: If the presentation contains 3 or more bytes, use text, otherwise use numerics.

Any characters in the pattern that are not in the ranges of [’a’..’z’] and [’A’..’Z’] will be treated as quoted text. For example, the following characters will appear in the resulting time text (even if they are not within single quotation marks):

- ’:’
- ’.’
- ’ ’
- ’#’
- ’@’

Patterns containing an invalid pattern letter result in a thrown exception during formatting or parsing.

A Time and Date Format Example

The examples in Table 19 use a US Location.

Table 19. Examples of time and date formatting.

Format Pattern	Result
`"yyyy.MM.dd G 'at' hh:mm:ss z"`	2001.04.01 AD at 15:08:56 PDT
`"EEE, MMM d, ''yy"`	Sun, April 01, ’01
`"h:mm a"`	12:08 PM
`"hh 'o''clock' a, zzzz"`	12 o’clock PM, Pacific Daylight Time
`"K:mm a, z"`	0:00 PM, PST
`"yyyyy.MMMMM.dd GGG hh:mm aaa"`	2001.April.01 AD 12:08 PM

Customizing Line Thickness

You can adjust the thickness of lines in a topology view. Table 20 shows the values for line thickness; these values are set in the defaultscheme.properties file.

Table 20. Changing line thickness (in pixels).

`defaultscheme.properties` Attribute	Default Value (in pixels)	Minimum Value (in pixels)	Maximum Value (in pixels)
line_thickness	2	1	4

To customize the line thickness, the NetView management console operational scheme must be updated, as shown in Table 20. This can be done from the console or the server workstation, depending on the mode in which you are running. See Table 7 on page 23 to determine if the updates should be performed from the server or console workstation. To customize line thickness, perform the following steps.

1. Make a backup copy of the default NetView management console operational scheme.
2. Open the `defaultscheme.properties` file in a text editor.
3. Search for the string `line_thickness`.
4. Change the value on this line to the desired line thickness (in pixels). The valid range is from 1 to 4.
5. Save the changes.

To verify this change, perform the following steps:

1. Start an NMC console.
2. Open a view.
3. Verify the line thickness while in the topology view.

Line Thickness Example

The example in Table 21 on page 34 shows the changes made to the line thickness from two pixels to one pixel.

Table 21. Changing the line thickness in the defaultscheme.properties file.

Before	After
************************ * Thickness of connection lines in pixels ************************ line_thickness = 2	************************ * Thickness of connection lines in pixels ************************ line_thickness = 1

For more information on the NetView management console operational scheme, see Table 6 on page 23.

Customizing Topology Server Features

You can customize features of the NetView management console topology server, by adding to or changing the server.properties file.

Customizing the Server Properties File

A server.properties file is provided with the topology server. This is a plain text configuration file that enables control of the following features:

- Communications time-outs
- Performance tuning
- View appearance

These features are used by the server at run time and affect all consoles attached to the server. Normally, these features should be left at the default levels. The file contains a detailed description of the items available for configuration. The server.properties file is located in one of the following directories:

- For Intel: `%BINDIR%\TDS\server\config`
- For UNIX: `$BINDIR/TDS/server/config`

To customize the topology server features, perform the following steps:

1. Create a back up of the server.properties file.
2. Open the server.properties file in a text editor and make the necessary changes.
3. Restart the topology server for the changes to take affect.

Chapter 5. Creating a Demo

You can create your own new NetView management console demo, customized to look and feel like your NetView management console environment and display views from your network environment. This section provides the process to create your own demo.

You can capture views from a live NetView management console system and integrate them into your demo. This enables you to demonstrate NetView management console in your own environment to provide operator training, for example.

The following sections contain the steps that are required to create a new demo.

1. "Capturing Live Views from your NetView Management Console System".
2. "Integrating Captured Views into the Demo" on page 42 by:
   a. "Updating the Business Tree" on page 42.
   b. "Renaming Navigation Views" on page 44.
3. "Defining New Resource Types in Saved Views" on page 44.

Once these steps are completed, the files shown in Table 22 will comprise the new demo.

Table 22. Demo files.

Demo	Directory	Files
View files (basic data files)	<installRoot>\bin\generic_unix\TDS\client\views	*.md
Basic Data definitions (See "Using Basic Data Files" on page 37 to define additional resource types).	<installRoot>\bin\generic_unix\TDS\client\settings	basicdata.nmc

Capturing Live Views from your NetView Management Console System

To capture views from a live NetView management console system for subsequent integration into the demo, use the `-saveViewsLocally` command line argument. This argument instructs the `Save View Customization` code to save the view in a local file in the <installRoot>\bin\generic_unix\TDS\client\views directory instead of within the topology server database.

Note: For more information about local files, see "Renaming Navigation Views" on page 44.

To capture a view from the Client workstation and to save all the views that you want to appear in your demo, perform the following steps.

1. From the command prompt, make a backup copy of the demo views shipped with NetView management console as follows:
   a. `cd <installRoot>\bin\generic_unix\TDS\client\views`
   b. `md backup`

c. `copy * backup`

2. From the command prompt, delete all of the shipped NetView management console demo files as follows:
   a. `cd <installRoot>\bin\generic_unix\TDS\client\views`
   b. `del *`
3. Start the NMC Console by using the `-saveViewsLocally` command line argument. To do this, change the Properties of the NMC Topology Console desktop icon, as follows:
   a. Right-click on the **NMC Topology Console** desktop icon to display the context menu.
   b. Click **Properties**.
   c. From the notebook that opens, select the **Shortcut** tab.
   d. Select the **Target** field and place the cursor after the last character.
   e. Type a separating blank and then enter **-saveViewsLocally**.
   f. Click **OK**.
4. Start the NMC console and sign on.
5. Save the business tree view, as follows:
   a. Right click on the background of the business tree.
   b. Select **Save View Customization**.

      This creates the <installRoot>\bin\generic_unix\TDS\client\views\1.ctl.md file. The root node of the business tree is always `resource ID 1`.

      Note: For more information on resource Ids, see “Finding the Resource ID” on page 45.
   c. Save all of the specific views to be included in your demo by opening each view individually. To open each view, do one of the following:
      - From the business tree, open each view sequentially.
      - Select **Locate Resource**.
      - Select **More Detail**, **Configuration**, or **Locate Failing Resource** from an open view.
   d. To save each view:
      1) Right click on the view background.
      2) Select **Save View Customization**.
   e. Correlate the type of view saved with the saved name by performing the following steps:
      1) Write down information such as the view type, view name, Configuration Parents, More Detail, and Locate Resource.
      2) Look in the <installRoot> \bin\generic_unix\TDS\client\views\ directory for the view that you just saved. It will have the format <resourceid>.ctl.md. Write down this file name next to the view type. You will need this information when you rename the view to conform to the demo naming conventions later.

         The view you just created will be the newest view in the directory. If you have kept track of each of the demo views as you created them, you should be able to distinguish which view was just created to associate it with the view type.

         Note: Multiple navigation views can be created from the same resource.

A fixed file name is created for each view saved based on the ID of the resource, regardless of how the view is opened. If you save a More Detail and a Configuration view initiated from the same resource, the same view file name is used. The last view saved is the available view. If you want to have multiple navigation views available from a single resource, you *must* use these steps:

a) Select the Nth navigation view.

b) Save the view.

c) Rename the saved view.

d) Repeat these steps for the next navigation view.

6. Integrate the captured views into the demo as follows.
   a. Update the business tree. For detailed information about performing this step, see "Updating the Business Tree" on page 42.
   b. Rename the navigation views to conform to the required demo naming conventions. For detailed information about performing this step, see "Renaming Navigation Views" on page 44.
7. Define any resource types in the saved views that were not previously defined.

Using Basic Data Files

Because Resource Object Data Manager (RODM), the topology server, and the CPE command definitions are not available when the NMC console is run in demo mode, the basic data properties files are used to define:

- Resource types
- Individual context menu items
- Sets of context menu items
- A set of view menu items (resource independent)
- A default set of real (resource dependent) resource menu items
- A default set of aggregate (resource dependent) resource menu items

When the console starts, the following files are loaded from the <installRoot>\bin\generic_unix\TDS\client\settings directory:

- basicdata[<NLS>] (These are the base definitions of resources and common menu items.)
- basicdata[<NLS>].nmc (These are additional NetView management console resource types and menu items.)

Before making any changes to these files, make backup copies of these files with one of the following methods:

- Use a packaging tool such as PKZip, WinZip, or tar.
- Create a backup directory and copy the installed files into it.

The following Japanese files can also be customized, using double byte character set (DBCS) characters, with conversion commands:

- basicdata_ja
- basicdata_ja.nmc
- defaultschemetext_ja.properties

To customize these files, first convert them with the following command prior to editing:

```
native2ascii -reverse <input_dir>\filename <outpug_dir>\filename
```

After editing, perform a second conversion with the following command:

```
native2ascii -encoding SJIS <input_dir>\filename <outpug_dir>\filename
```

Defining a Menu Item

Table 23 gives a description of the attributes available for menu item definition where the `N` in the attribute is a consecutive sequence number (1, 2, ..., n) within the group of menu items. Several types of menu items are supported. Each type has a separate definition list in the basic data file and different definition requirements, as shown in Table 24.

Table 23. Description of menu item attributes.

Attribute	Purpose
mi.N.<attr>	Java applications launched using `com.tivoli.ihs.client.action.IhsJavaAppAction`. **Note:** Use this type of attribute for testing Java applications.
sc.N.<attr>	Predefined scenarios.
nop.N.<attr>	NOP menus which appear in the context menu but do not do anything when selected.

Table 24. Attributes that can be specified for a menu item.

<attr>	Description	mi.N.<attr>	Sc.N.<attr>	nop.N.<attr>
tag	*Tag* name that is used to reference this menu item	Required	Required	Required
label	Menu item *text*	Required	Required	Required
class	Package-qualified Java class that provides the Java implementation of this menu item	Optional Def: IhsJavaAppAction[1]	Ignored Def: IhsScenario[1]	Ignored Def: IhsCPEAction[1]
data	Class specific data	Optional Def: blank	Required scenario file name and arguments	Ignored
html	**Menu Help** HTML panel name for the menu item	Optional Def: ihs_nohelp_XXX.html	Optional Def: ihs_nohelp_XXX.html	Optional Def: ihs_nohelp_XXX.html
max	Maximum number of selected resource(s) supported by a particular resource type	Optional Def: 1	Optional Def: 1	Optional Def: 1
debug	Determines if this is a debug menu item; enabled when the -debug command line argument is used	Optional Def: false	Optional Def: false	Optional Def: false

Table 24. Attributes that can be specified for a menu item. (continued)

<attr>	Description	mi.N.<attr>	Sc.N.<attr>	nop.N.<attr>
vfy	Determines if a verification prompt should appear before item is launched[2]	Optional Def: false	Optional Def: false	Optional Def: false

Notes:

1. These classes are part of the com.tivoli.ihs.client.action package.
2. This attribute, <attr>, is not currently implemented.

Defining a Set of Menus

A set of menus is a named group of one of the following:

- Menu items
- Separators
- Other sets

A set can be used to define either of the following:

- A set of menu items that can be referenced as many times as needed
- A cascaded menu item

Conditional debug sets are enabled when the -debug command line argument is used.

A set is defined using the following syntax:

```
set.N.<attr>=value
set.N.X     =tag reference
```

where:

- The set stem uniquely identifies this as a set definition.
- N is a consecutive sequence number (1, 2, ..., n) within the set definition area.
- X is a consecutive sequence number (1, 2, ..., n) within a set.

The attributes in Table 25 can be specified for a set.

Table 25. Attributes that can be specified for a set.

<attr>	Description	Default
tag	Name used to reference this set	Required
label	Name of cascaded menu item	Optional • If omitted, the set is used as a container; items are copied into the current menu. • If specified, a cascaded menu item is created with the menu items for this set.
debug	Determines if this is a debug set; enabled when the -debug command line argument is used	Optional Def: false

Each item of a set references one of the items in Table 26.

Table 26. Set reference items.

Tag Reference	Description
- (dash)	Menu separator
`tag` name	Reference to a: • menu item • set

A Sample from basicdata.nmc: The following is an example from the basicdata.nmc file.

```
################################################
# Context menu items for an AGGREGATE resource
################################################
set.3.tag   =@nmcAgg
set.3.1     =@baseAgg
set.3.2     =-
set.3.3     =nmcFailing
################################################
# "Configuration" cascaded menu
################################################
set.4.tag   =@nmcConfig
set.4.label =Configuration
set.4.html  =ihs_mi_config_XXX.html
set.4.1     =nmcCfgPar
set.4.2     =nmcCfgChild
set.4.3     =nmcCfgPeer
set.4.4     =nmcCfgLP
set.4.5     =nmcCfgL
set.4.6     =nmcCfgP
set.4.7     =nmcCfgBBone
################################################
# SET for all example Java applications
################################################
set.5.tag   =@exJavaApps
set.5.debug =true
set.5.1     =-
set.5.2     =exGUI
set.5.3     =exCmdRsp
set.5.4     =exIPL
```

Defining a Resource Type

A resource type is defined using syntax as shown in Table 27.

Table 27. Attributes that can be specified for a resource type.

<attr>	Description	Default
rtX_name	Name; specified in node/link ″resource type″ field	
rtX_desc	Description text	Optional resource type name
rtX_image	Image file name	Optional node.gif
rtX_geometric	Geometric shape name	Optional com.tivoli.ihs.reuse.gui.IhsRectangle

Table 27. Attributes that can be specified for a resource type. (continued)

<attr>	Description	Default
rtX_help	Help URL	Optional none
rtX_flags	Flags	Optional 1

The following is an example from the basicdata.nmc file:

```
rt34_name=268828673
rt34_desc=LU
rt34_image=duiu5n00.gif
rt34_geometric=com.tivoli.ihs.reuse.gui.IhsTrapezoid
```

You can also specify additional context menu items for specific resource types. In the following example, each `rtN_mi.X` item is a tag reference.

```
rt6_name=Lotus Notes Client
rt6_image=notesc.gif
rt6_geometric=com.tivoli.ihs.reuse.gui.IhsPentagon
rt6_mi.1=-
rt6_mi.2=start
rt6_mi.3=stop
rt6_mi.4=busSysHelp
rt6_mi.5=instSmtpGW
```

The current syntax for defining a resource type is available by using the following command when `CLASSPATH` includes the ihseuc.jar file:

```
java com.tivoli.ihs.client.view.IhsResourceType
```

Defining Defaults

Each basic data file defines defaults for the following:

- View context menu items
- Common context menu items for each real resource
- Common context menu items for each aggregate resource

These defaults are shown in the following example:

```
##########################################################
# Default VIEW items
#
# - each SET reference is a "tag"
# - these values override those in "basicdata"
##########################################################
view.addDefault=true            <-- controls if defaultSet is added
view.defaultSet=@nmcView        <-- set tag reference

##########################################################
# Default Resource Type items for this "basic data" file
#
# - each SET reference is a "tag"
# - these values override those in "basicdata"
##########################################################
agg.addDefault =true            <-- controls if defaultSet is added
agg.defaultSet =@nmcAgg         <-- set tag reference

real.addDefault=true            <-- controls if defaultSet is added
real.defaultSet=@nmcReal        <-- set tag reference
```

Creating a New Resource Type

To create a new resource type, do the following:

1. Add the required `rtN_xxx` entries at the end of the appropriate basicdata file.

   Note: The sequence numbers, the `N` part of `rtN`, must be sequential with no intervening gaps.

2. To define the geometric shape displayed on a topology view, specify the appropriate package qualified class names as shown in the following list:
   - com.tivoli.ihs.reuse.gui.IhsCircle
   - com.tivoli.ihs.reuse.gui.IhsCircleStar
   - com.tivoli.ihs.reuse.gui.IhsDiamond
   - com.tivoli.ihs.reuse.gui.IhsHexagon
   - com.tivoli.ihs.reuse.gui.IhsOctagon
   - com.tivoli.ihs.reuse.gui.IhsOval
   - com.tivoli.ihs.reuse.gui.IhsParallelogram
   - com.tivoli.ihs.reuse.gui.IhsPentagon
   - com.tivoli.ihs.reuse.gui.IhsRectangle
   - com.tivoli.ihs.reuse.gui.IhsRoundRect
   - com.tivoli.ihs.reuse.gui.IhsSolidLine
   - com.tivoli.ihs.reuse.gui.IhsStar
   - com.tivoli.ihs.reuse.gui.IhsTrapezoid
   - com.tivoli.ihs.reuse.gui.IhsTriangle

Example: `rtN_geometric=com.tivoli.ihs.reuse.gui.IhsHexagon`

To use the new resource type in a view, set the `anN_t` attribute of a resource to the `rtN_name` attribute of your new resource type. For example: `an1_t=2147614793`.

Integrating Captured Views into the Demo

Updating the Business Tree

After the business tree has been saved, some work is required to make it usable by the NetView management console code in demo mode. Note that the original business tree file, shipped with NetView management console is the avail.control.md.nmc file. It is located in the <installRoot>\bin\generic_unix\TDS\client\views\backup directory you created before saving views. This file illustrates the parent-child relationships used within the business tree to create the tree node, branches, and leaf nodes.

To make the saved business tree available, perform the following steps from the client workstation.

1. From the command prompt, rename the saved business tree view as follows:
   a. `cd <installRoot>\bin\generic_unix\TDS\client\views`
   b. `copy 1.ctl.md  avail.control.md.nmc`

   Notes:

   a. The root node of the business tree is always `resource ID 1`. For more detailed information about the resource id, see “Finding the Resource ID” on page 45.

b. Copying the file, instead of renaming it, enables you to keep the original if you want to restart the process during this procedure.

2. Edit the avail.control.md.nmc file.
3. Replace the complete view model at the top of the file with the line in the **After** column in Table 28. For more detailed information about the view model, see "Defining View Information" on page 48.

Table 28. Replacing the view model text example (before and after).

Before	After
view_id=xxxx	view_id=availAvail
view_prev_id=-1	view_width=500
view_useimage=-1	view_height=900
view_openview=0	
view_width=xxxx	
view_height=xxxx	
view_cust=true	
view_customizable=true	

4. Remove all parent references to the root node of the tree by doing the following:
   a. Locate every parent reference by searching for the string `_p=1`.
   b. Delete each of these lines containing `_p=1`.
   c. Verify that the value is `1` (and not `1x` or `1xx`, for example).
5. Update the resource type for every node by inserting the `RT` prefix as shown in Table 29.

Table 29. Updating the resource type example (before and after).

Before	After
anx_t=536871171	anx_t=**RT**536871171

6. Verify that the special business tree nodes shown in Table 30 are defined to use these specific resource type values. If a different value is present as the `anX_t` attribute, replace it with the required value shown in Table 30.

   This enables you to avoid error messages when NetView management console is started and have the correct icon displayed in the business tree.

Table 30. Verifying the resource type values for business tree nodes.

Special Business Tree Node (found in the anX_I attribute)	Required Resource Type (defined in basicdata.nmc)
Business Systems	RT536871171
Systems Management Business System	RT536871228
Networking	RT536871168
Network Views	RT536871169
Exception Views	RT536871170

7. Remove the following attributes from every node in the business tree:
   - `anx_da=true`
   - `anx_def=true`
8. Save your changes.

To verify these changes, do the following:

a. Start the NetView management console Demo.
b. Verify that the business tree is properly displayed.
c. Verify that the views you saved, which were initiated from the business tree, will open.

9. Integrate the captured views into the demo by renaming the navigation views to conform to the naming conventions for your demo.
10. Define any resource types in the saved views that were not previously defined.

Renaming Navigation Views

To rename the navigation view, first locate the correct navigation file. All of the target navigation files are located in the <installRoot\bin\generic_unix\TDS\client\views> directory. Table 31 shows the naming conventions.

Table 31. Naming conventions for navigation views.

Navigation Context Menu	Target File Naming Convention
Configuration > Parents	<resource ID>.par.ctl.md
Configuration > Children	<resource ID>.chd.ctl.md
Configuration > Peers	<resource ID>.peer.ctl.md
Configuration > Logical and Physical	<resource ID>.lp.ctl.md
Configuration > Logical	<resource ID>.log.ctl.md
Configuration > Physical	<resource ID>.phy.ctl.md
Configuration > Backbone	<resource ID>.bak.ctl.md
Locate Failing Resources	<resource ID>.fp.ctl.md
Locate Resource	<locate name>.locate.md
More Detail	<resource ID>.ctl.md

Defining New Resource Types in Saved Views

The views you save might contain resources that are not currently defined for the demo. This section provides the steps necessary to define missing RODM resource types to the demo. When a resource type is referenced in a view that is not currently defined, messages similar to the following are written to `stderr` when the view is opened.

```
IhsViewModel:verifyView(save) Thread-8 p1=64 p2=ResourceType (2416050177) specified
on 2 was not found
IhsViewModel:verifyView(save) Thread-8 p1=64 p2=Setting 2's resource type to IhsNodeRT
IhsViewModel:verifyView(save) Thread-8 p1=1085 p2=ResourceType (2147549291) specified
on 3 was not found
IhsViewModel:verifyView(save) Thread-8 p1=1085 p2=Setting 3's resource type to IhsLinkRT
```

Where:

- `2416050177` is the resource type specified in the `anX_t` attribute of a node resource.
- `2` is the resource ID specified in the `anX_r` attribute of a node resource.
- `2147549291` is the resource type specified in the `alXt` attribute of a link resource.
- `3` is the resource ID specified in the `alX_r` attribute of a link resource.

In a view, these resources display as icons.

All of the information required to define a RODM resource type is available from the Legend window when you are connected to a live NetView management console system. To define missing resource types referenced by saved views from the client workstation, perform the following steps.

1. Select **Help→Legend...** to open the Legend window.
2. To locate the resource type, search the ID column of the Legend window for a decimal value (2416050177, for example). This is the resource type that must be added.
3. Edit the basicdata.nmc file.

   For more information on this file, see "Using Basic Data Files" on page 37.
4. Using the information in Table 32, create a new resource type at the end of the currently defined set.

Table 32. Creating a new resource type from the management console – Legend window.

Resource Type Attribute	Description	Legend Column
rtX_name	Name	use the decimal ID value (2nd value)
rtX_desc	Description text	Resource Type
rtX_image	Image file name	Icon File Name
rtX_geometric	Geometric shape	map shape to the corresponding geometric
rtX_help	Help	Help File Name
rtX_flags	Flags	Flags

   For more information on creating new resource types, see "Creating a New Resource Type" on page 42.

   The following example shows how an APPN® end node would be defined:

```
rtX_name=327883
rtX_desc=APPN end node (EN)
rtX_image=duiu5n00.gif
rtX_geometric=com.tivoli.ihs.reuse.gui.IhsOctagon
```

5. Save your changes.
6. Start the NetView management console Demo and verify that the proper icon is now displayed.

Finding the Resource ID

The topology server assigns each resource an internal resource ID value. This value is important for demo mode because it is used to generate the name of navigation view files. To determine the ID of a resource on a live NetView management console System, perform the following steps.

1. Sign on.
2. Select the **Options** menu item and then **Console Properties...** to open the **Console Properties** notebook.
3. Select the **Service** tab, which is the far right tab in the notebook. If it is not visible, scroll to it.
4. Check **Action** in the **Components** section.
5. Check **Debug** in the **Trace types** section.
6. Click **OK**.

7. Select the view that contains the resource ID you want to find.
8. Left click the resource to open the context menu for that resource and select **Resource Properties**.
9. Select the **Debug** tab. The Resource Properties window will display debug data about the resource.
10. Look for `ibmId` in the `property` field. The corresponding `Value` is the ID for the resource. The ID value is displayed in decimal and hexadecimal. Usually, you will need to use the decimal value.

Defining a Node Resource in a View

The attributes used to define a node resource in a view are shown in Table 33.

Table 33. Attributes to define a node resource in a view.

Attribute	Definition
anX_i	Display ID (unique per view).
anX_r	Server ID (unique per server); double clicking the resource drills to <serverID>.md.
anX_p	Parent ID (optional, none).
anX_t	Resource type (optional, defaulted).
anX_s	Status (optional, normal).
anX_u	Flags (optional, zero).
anX_a	Determines if this is an aggregate resource (optional, false).
anX_da	Suppress '+' on aggregate resources. For Topology Display Subsystem view only, others not *live* if suppressed (optional, value of anX_a).
anX_l	Label (optional, blank).
anX_lx	Label X coordinate (optional, auto).
anX_ly	Label Y coordinate (optional, auto).
anX_d1	Data 1: TDS=HB1, NMC=RODM other data (optional, blank).
anX_d2	Data 2: TDS=HB2, NMC=RODM customer data (optional, blank).
anX_d3	Data 3: TDS=HB3, NMC=not used (optional, blank).
anX_d4	Data 4: TDS=HB4, NMC=not used (optional, blank).
anX_per	On the business tree view, determines if a view is permanent (optional, false).
anX_cag	Specifies if customer aggregation is allowed (Not currently in use.) (optional, true).
anX_c1	Monitor Count 1 (optional, none).
anX_c2	Monitor Count 2 (optional, none).
anX_x	X coordinate.
anX_y	Y coordinate.

In the documenting class, `com.tivoli.ihs.client.view.IhsNode`, the following example shows how a node resource can be defined:

```
an3_i=3
an3_r=2100
an3_p=39
an3_t=327882
an3_l=NETA.FVT01EM
```

```
an3_s=18
an3_per=false
an3_x=342
an3_y=106
```

Defining a Link Resource in a View

The attributes used to define a link resource in a view are shown in Table 34.

Table 34. Attributes to define a link resource in a view.

Attribute	Definition
alX_i	Display ID (unique per view).
alX_r	Server ID (unique per server); double clicking the resource drills to<serverID>.md.
alX_p	Parent ID (optional, none).
alX_t	Resource type (optional, defaulted).
alX_s	Status (optional, normal).
alX_u	Flags (optional, zero).
alX_a	Determines if this is an aggregate resource (optional, false).
alX_da	Suppress '+' on aggregate resources. For Topology Display Subsystem view only, others not ″real life″ if suppressed (optional, value of alX_a).
alX_l	Label (optional, blank).
alX_lx	Label X coordinate (optional, auto).
alX_ly	Label Y coordinate (optional, auto).
alX_d1	Data 1: TDS=HB1, NMC=RODM other data (optional, blank).
alX_d2	Data 2: TDS=HB2, NMC=RODM customer data (optional, blank).
alX_d3	Data 3: TDS=HB3, NMC=not used (optional, blank).
alX_d4	Data 4: TDS=HB4, NMC=not used (optional, blank).
alX_per	On the business tree view, determines if a view is permanent (optional, false).
alX_cag	Specifies if customer aggregation is allowed (Not currently in use.) (optional, true).
alX_c1	Monitor Count 1 (optional, none).
alX_c2	Monitor Count 2 (optional, none).
alX_1	First end point able to be displayed.
alX_2	Second end point able to be displayed.
alX_d	Link direction with respect to the first end point • 0=none (default) • 1=origin • 2=destination • 3=bidirectional • 4=replica

In the documenting class, `com.tivoli.ihs.client.view.IhsLink`, the following example shows how a link resource can be defined:

```
al15_r=268
al15_p=216
al15_l=40001A20AC05
```

```
al15_d1=Bridge=3F10TOP, Segment=020A, MAC Address=40001A20AC05
al15_per=false
al15_1=28
al15_2=9
```

Defining View Information

Table 35 shows the attributes used to define the appearance of a view.

Table 35. Attributes to define appearance of a view.

Attribute	Definition
view_id	ID for the view
view_prev_id	Previous ID for the view
view_width	Width of view (optional, 500)
view_height	Height of view (optional, 300)
view_layer	Layer of view
view_descriptor	Descriptor of view (optional, "")
view_fgcolor	Foreground text color of view; RGB value (optional, black)
view_ftcolor	Free text color of view; RGB value (optional, black)
view_useimage	Specifies whether to use image(1)/color(0)/notSet(-1) for view background (optional, -1)
view_bgcolor	If color, the background color RGB value (optional, gray)
view_bg	If image, specific image file name
view_bgx	If image, image X location within view (optional, -1)
view_bgy	If image, image Y location within view (optional, -1)
view_bgwidth	If image, width (optional, -1)
view_bgheight	If image, height (optional, -1)
view_swidth	Width of resource symbol area (optional, 40)
view_sheight	Height of resource symbol area (optional, 40)
view_cust	Specifies if the view is customized (optional, false)
view_customizable	Specifies if the view can be customized (optional, true)
view_ffu	Specifies whether or not to force full update of view (optional, false)
view_orp	Specifies if override refresh property (optional, false)
view_activetab	View automatically opened for tab (optional, first tab)
view_openview	Open view option3 (topology or detail view)#

In the documenting class, `com.tivoli.ihs.client.view.IhsViewModel`, the following example shows how the appearance of a view can be defined:

```
al15_r=268
al15_p=216
al15_l=40001A20AC05
al15_d1=Bridge=3F10TOP, Segment=020A, MAC Address=40001A20AC05
al15_per=false
al15_1=28
al15_2=9
```

Defining a Demo View

A demo view is an ASCII flat file that is typically created as described in “Capturing Live Views from your NetView Management Console System” on page 35. It contains the following collection of object definitions:

1. A view model definition, as described in “Defining View Information” on page 48.
2. Zero or more node definitions, as described in “Defining a Node Resource in a View” on page 46.
3. Zero or more tackpoint definitions.
4. Zero or more free text definitions.
5. Zero or more link definitions, as described in “Defining a Link Resource in a View” on page 47.

The following example shows how a demo view can be defined.

```
################################################################################
# View Definition File
#
# Resource: 89
#
# Created:  Mon Feb 01 10:36:29 PST 1999
#
# Warning:  Be careful if you modify this by hand!
################################################################################
view_id=89
view_prev_id=-1
view_bgcolor=-1
view_fgcolor=-16777216
view_ftcolor=-16777216
view_useimage=1
view_openview=0
view_width=398
view_height=240
view_cust=true
view_customizable=true
view_descriptor=NETA.4-MDL
################################################################################
# Node objects
################################################################################
an1_i=1
an1_r=473
an1_p=461
an1_t=2147811538
an1_l=D:NETA.NRILOV00
an1_u=134217728
an1_a=true
an1_da=true
an1_per=false
an1_x=200
an1_y=60
an2_i=2
an2_r=474
an2_p=461
an2_t=327776
an2_da=true
an2_per=false
an2_x=200
an2_y=180
################################################################################
# Tackpoint objects
################################################################################
################################################################################
# Free Text objects
```

```
#############################################################################
#############################################################################
# Link objects
#############################################################################
al1_i=3
al1_r=475
al1_p=461
al1_l=IC:NETA.NRILOV00.USIBMNT.NTFEMVS
al1_s=20
al1_a=true
al1_da=true
al1_per=false
al1_1=1
al1_2=2
```

Chapter 6. Topology Console Java Applications and Plug-ins

This chapter describes the Java applications and plug-ins provided with the NetView management console. You can write applications or plug-ins to enhance topology console operation.

The example Java applications and plug-ins are installed as part of the *NetView management console Productivity Kit* (on the Intel platform, only). Be sure to perform a custom install of the NMC Console. Because the technical information about these applications and plug-ins is dynamically created using javadoc, the applications and plug-ins cannot be described in their entirety. References will be made here to these examples in the *NetView management console Productivity Kit*. The *NetView management console Productivity Kit* contains more detailed technical information (such as methods, fields, syntax and Java class hierarchy). To find this information, access the *NetView management console Productivity Kit*, in one of the following ways:

- From Windows, double-click the **NMC Productivity Kit** icon on the desktop.
- From OS/2, open the **NMC Topology Console** folder and double-click **NMC Productivity Kit**.

Note: Though the *NetView management console Productivity Kit* can be installed only on the Intel platform, the functionality of Java applications and pug-ins is available on any supported NMC Console platform.

Supplied Support Files

The files shown in Table 36 provide additional support for the Java application and Java plug-in examples.

Table 36. Files that support the Java examples.

File	Description
examples\java\ExampleJavaApp.jar	This Java ARchive (JAR) file contains the compiled class files of the examples, so they are immediately usable.
examples\support\ExampleJavaApp.rsp	This server Command Profile Editor (CPE) response file is used to create the context menu command definitions for the example Java applications so that the server will include them in context menus.
examples\support\plugins.properties	This is a plug-in definition file that loads all of the example plug-ins.
examples\javadoc	This is the starting point for the console API documentation.

To access these files, see the *Supplied Support Files* section of the *NetView management console Productivity Kit*.

Installing the Examples

The examples and support files are installed as part of the *NetView management console Productivity Kit*. Currently, the *NetView management console Productivity Kit* can be installed only on the Windows and OS/2 platforms. See the Supplied Support Files section of the *NetView management console Productivity Kit* to view an example installation structure.

Enabling the Examples

Although the examples have been installed, certain steps are required to enable their use. The exact steps vary, depending on how you want to access the examples. You can access the examples either while connected to a server or while running the console in demo mode.

To run the examples while signed on to a server, perform these steps:

1. From the server workstation, access the console workstation where you have installed the example files.
2. Change to the `lib` directory: `cd %BINDIR%\TDS\server\db\current\lib` where the `BINDIR` environment variable defines the installation root of the server.
3. Copy the <console machine install path>\bin\generic_unix\TDS\client\examples\java\ExampleJavaApp.jar to this directory. The ExampleJavaApp.jar file will now automatically be downloaded to each console that subsequently signs on to this server.

To run the examples in demo mode, perform these steps:

1. From the console workstation, change to the <install path>\bin\generic_unix\TDS\client\examples\support directory.
2. Issue the following command:

   ```
   copy plugins.properties ..\..\settings
   ```

Compiling the Examples

It is not necessary to compile the examples before they are used. A provided JAR file, examples\java\ExampleJavaApp.jar contains the compiled examples. See Table 36 on page 51 for a brief description of this file.

If you change an example, you will need to recompile the example. Before you recompile the examples do the following from a console workstation:

1. Install the console code.
2. Install Java™2 SDK, Standard Edition, v1.3 (J2SDK).

To recompile the examples after you have changed them, do the following:

1. Change to the <install path>\bin\generic_unix\TDS\client\examples\java directory.
2. Compile the examples by issuing the following command:

   ```
   javac -classpath %CLASSPATH%;..\..\lib\ihseuc.jar *.java
   ```

   Note: To compile cleanly, the `CLASSPATH` environment variable must include the console code and the J2SDK classes.
3. Create a new JAR file (to contain the compiled classes) using the following command:

   ```
   jar -cfv ExampleJavaApp.jar *.class
   ```

4. Make the new JAR file available. For more information on this process, see “Enabling the Examples” on page 52.

Tracing the Examples

The examples have been instrumented with RAS tracing. You can enable this tracing from the Service page of the Console Properties notebook. To enable tracing, do the following:

1. Select the **Options** menu item and then **Console Properties...** to open the **Console Properties** notebook.
2. Select the **Service** tab, which is the far right tab in the notebook. If it is not visible, scroll to it.
3. Select the **Customer** component.
4. In the Trace types section, click one or more of the following check boxes:
   - **Constructors**
   - **Public methods**
   - **Callback methods**
5. In the Additional Tracing Controls section, check the **Details** box so that all of the traced data is displayed.
6. Click **OK**.

Problem Determination

Message `IHS1011W` is used to report problems encountered while loading a plug-in. Because plug-ins are loaded before the console window has been created, this message cannot be displayed in a pop up window or recorded in the Log window. Therefore, it is written to `stderr`.

The following errors are reported in the `Why:` field that is specific to the plug-in:

The specified plug-in class does not exist

```
WARNING: Could not instantiate bean "PlugIn.Does.Not.Exist"
from JAR "d:\Tivoli\bin\w32-ix86\..\generic_unix\TDS\client\lib\ExampleJavaApp.jar"
    We couldn't open the class file "PlugIn/Does/Not/Exist.class" in the JAR
IHS1011W: Unable to start a Java application.
Class: PlugIn.Does.Not.Exist
Why: Plug-in class not found
Phase: 1
```

Note: The first 3 lines of the preceding example are generated by the dynamic Java class loading mechanism of the console.

The specified plug-in class exists but does not implement the IhsIPlugIn interface

```
IHS1011W: Unable to start a Java application.
Class: com.tivoli.ihs.client.IhsClientArgs
Why: Plug-in does not implement IhsIPlugIn interface
Phase: 2
```

Java Applications

A Java application consists of customer-written code that is initiated from a context menu item (as a context menu item). The Application is defined in the command profile editor and runs on the topology console (a Java virtual machine).

The context menu is defined to the topology server using one of the following:

- command profile editor (CPE) user interface program

- `CPEBATCH` batch utility

Java applications can be either resource dependent or independent. Dependent Java applications provide information about each selected resource. The `com.tivoli.ihs.client.action.IhsResInfo` class, for example, provides this information. Java applications provide information about their associated context menu items. The `com.tivoli.ihs.client.action.IhsCmdInfo` class, for example, shows this. A Java application extends the `IhsJavaApplicationAdapter` class.

Once launched, the application can access any services provided by Java, certain topology console services, or any additional customer or third party services. A command can be generated and issued using any of the command exits provided by the topology server. For more information about command exits, see “Chapter 9. Using the NMC Command Profile Editor” on page 87.

Note: See the *Java Applications* section of the *NetView management console Productivity Kit* For more details about any of the following:

- The `com.tivoli.ihs.client.action.IhsResInfo` class
- The `com.tivoli.ihs.client.action.IhsCmdInfo` class
- The `IhsJavaApplicationAdapter` class
- Topology console services

Java Application Examples

The example Java applications shown and described in Table 37 are provided with NetView management console.

Table 37. Available Java application examples and descriptions of each.

File	Description
examples\java\ExampleGUIJavaApp.java	Displays context in a window. It shows all of the available information about the selected resources. It can be used as either a resource dependent or independent command.
examples\java\ExampleCmdWithResponse.java	Illustrates sending a command to the Tivoli NetView for OS/390 program and receiving all response lines for subsequent processing.
examples\java\ExampleGUItoIML.java	Illustrates writing a complex graphical interface. It displays the data required to start a 3174 device and enables the user to select appropriate options. A command is constructed from the user input and sent to the Tivoli NetView for OS/390 program. The results from the command are displayed in the console Log window.

Note: For detailed information about these Java classes, see the *Java Application Examples* section of the *NetView management console Productivity Kit*.

Java Application Development Process

The software development steps used when developing a Java application are as follows:

1. Edit

a. Use one of the provided Java application examples as a starting point.
b. Customize the example to meet your specific requirements.

2. Compile, Package, and Deploy your application. For more information on this process, see “Compiling the Examples” on page 52.
3. Enable the new Java application. For more information on this process, see “Defining the Example Java Applications”.
4. Test, as follows:
   a. Sign on to the server containing the new code.
   b. Display the Java application in a context menu.
   c. Select the menu item and verify that it is working properly.
   d. If necessary, enable service tracing options to help diagnose problems. For more information on tracing, see “Tracing the Examples” on page 53.

Defining the Example Java Applications

From the Server

To run the example Java applications while signed on to a server, do the following:

1. Go to the command prompt of a workstation on which the NetView management console server and console are both installed.
2. Use the server `CPEBATCH` program to define the example Java application so that it appears in the context pop up menu for a resource.
3. Change to the bin directory (where the `BINDIR` environment variable defines the installation root of the server): `cd %BINDIR%\TDS\server\bin`.
4. Issue the following command:

```
cpebatch <console machine install path>\bin\generic_unix\TDS\client\examples
\support\ExampleJavaApp.rsp -i -g
```

In Demo Mode

To run the application examples in demo mode, from the console workstation, do the following:

1. Change to the `<install path>\bin\generic_unix\TDS\client\settings` directory.
2. Edit the basicdata.nmc file, locating the `set.5.debug` key (which is part of the `@exJavaApps` group) and changing the value from `true` to `false`.
3.

Running the Example Java Applications

From the Server

To run the example Java applications from the server, do the following:

1. Start the console using the standard desktop icon.
2. Sign on to the appropriate server.
   - For resource independent examples, do the following:
     a. Position the cursor over any white space in the business tree.
     b. Right-click to display a context menu.
     c. Select an example menu item.

     Note: The context menu items for the example Java applications are named `Example: <application>`.
   - For resource dependent examples, do the following:

a. Open a view that contains at least one real resource.
b. Right-click the real resource to display a context menu.
c. Select an example menu item.

Note: The context menu items for the example Java applications are named `Example: <application>`.

In Demo Mode

To run the example Java applications in demo mode, do the following:

1. Start the console in demo mode using the standard desktop icon.
   - For resource independent examples, do the following:
     a. Position the cursor over any white space in the business tree.
     b. Right-click to display a context menu.
     c. Select an example menu item.
   - For resource dependent examples, do the following:
     a. Select **Tasks → Locate Resource** to open the Locate Resource window.
     b. Enter `NTFFPU20` in the entry field and then click the **Locate** button.
     c. In the view that opens, position the mouse over one of the nodes.
     d. Right-click to display a context menu.
     e. Select an example menu item.

Java Plug-Ins

Plug-in code enables you to control various aspects of console operation. Like a Java application, a plug-in is also Java code that runs within the topology console Java virtual machine, but differs from a Java application in that plug-ins:

- Are not related to any view or resource.
- Are loaded each time you sign on to a topology server (once the plug-ins definition properties file is downloaded from the server). For more information on the definition properties file see “Plug-In Definitions File” on page 58.
- Remain loaded and active as long as the operator is signed on.
- Have call back methods that are driven as many times as necessary.

Supported Plug-Ins

The topology console currently supports the following plug-ins:

- View Label Formatter Plug-in
- Log Window Filter Plug-in

View Label Formatter Plug-In

The view label formatter plug-in enables the programmatic control of the label text displayed on a topology view, a details view, or selected data windows. This plug-in must implement the IhsIPlugInViewLabel interface. To see detailed specifications, see the *Java Plug-ins* section of the *NetView management console Productivity Kit*.

Built-in console function provides some control over the displayed label text (for example, it might be truncated on the left or right), but the number of displayed characters cannot be controlled by this console function. This plug-in can be useful if you are adhering to resource naming conventions that include fixed prefixes or suffixes.

Some possibilities for customization with this plug-in include the following:

- Displaying the first *N* characters (where *N* is a number you specify)
- Removing common prefix or suffix text
- Combinations of removing and displaying text

The following are examples of data windows that use this plug-in when displaying resource names:

- Command for a Multi-owned Resource
- Event Viewer
- List Suspended Resources
- Session Data
- Status History

Note: The Resource Properties window intentionally does *not* use this plug-in so that the full resource name is always available.

Log Window Filter Plug-In

Commands, their generated responses, and console generated messages are centrally collected and displayed in the console Log window. The log window filter plug-in enables the programmatic control of the Log window contents. This plug-in must implement the `IhsIPlugInLog` interface. For detailed information about the `IhsIPlugInLog` interface, see the *Java Plug-ins* section of the *NetView management console Productivity Kit*. A new Log window entry can be handled as follows:

- It can be added as is.
  - The standard Log window display color is automatically used.
- It can be added with changes:
  - The text can be modified.
  - An override to the standard Log window display color can be specified.
- It can be suppressed.

You can also use this plug-in to initiate customer specific processing that results from a particular command.

Additional Plug-In Support

To determine which plug-ins are loaded, see the Environment Information window. The information shown in Table 38 is displayed for each active plug-in. For more information, see the *Java Plug-Ins* section of the *NetView management console Productivity Kit*.

Table 38. Values for plug-ins as shown in the Environment Information window.

Field Column	Value Column
<plug-in class name>:data	Initialization data passed to the `setPlugInData()` method.
<plug-in class name>:desc	Description as provided by the `getPlugInDescription()` method.
<plug-in class name>:version	Version as provided by the `getPlugInVRM()` method.

Table 38. Values for plug-ins as shown in the Environment Information window. (continued)

Field Column	Value Column
<plug-in class name>:debug	Debug information as provided by the `toString()` method. **Note:** If the string returned by `toString()` contains dynamic information, the Refresh button can be used to update this value without closing the window. The example plug-ins provide a template for this.

Notes:

1. The `-noPlugin` command line argument overrides automatic plug-in loading as each console invokes the plug-in.
2. When specified, the plug-in properties file is not processed after sign on.

Plug-In Definitions File

The plugins.properties file defines the console plug-ins that are to be loaded. This file is downloaded, after sign on, from the db\current\settings directory of the server.

Syntax for the plugins.properties file is shown in the following example:

```
*****************************************************************************
* Define the plug-ins to be loaded during Console initialization
* for every console that signs on to this server
*****************************************************************************
plugin.1.class = required full package qualified class name
plugin.1.data  = optional data passed to the setPlugInData() method for runtime use
plugin.2.class = <another>
plugin.3.class = <another>
...
```

Plug-In Examples

The example Java plug-ins shown in Table 39 can be used to enhance base console operation.

Table 39. Java plug-in examples.

File	Description
examples\java\ExampleLogPlugIn.java	Provides control over items (commands, responses, or messages) that are added to the console Log window Also see “Log Window Filter Plug-In” on page 57.
examples\java\ExampleViewLabelPlugIn.java	Provides control over the label that is displayed for a resource Also see “View Label Formatter Plug-In” on page 56.

Notes:

1. To use these example plug-ins to meet your specific requirements, specify the parameters in the data definition of the appropriate plug-in.
2. More information about all of the following is available in the *Java Plug-Ins* section of the *NetView management console Productivity Kit*:
   - Log window filter

- `com.tivoli.ihs.extern.plugin.IhsIPlugInLog` interface
- View label formatter
- `com.tivoli.ihs.extern.plugin.IhsIPlugInViewLabel` interface
- `ExampleLogPlugIn` class
- `ExampleViewLabelPlugIn` class

Plug-In Development Process

To develop a plug-in, use the following steps:

1. Edit an example, as follows:
   a. Use one of the provided plug-in examples as a starting point.
   b. Customize the example to meet your specific requirements.
2. Compile, package, and deploy your plug-in. For more information on this process, see “Compiling the Examples” on page 52.
3. Enable the plug-in. For more information on this process, see “Defining the Example Java Plug-Ins”.
4. Test the plug-in, as follows:
   a. Sign on to the server where the plug-in code resides.
   b. From the Environment Information window, do the following:
      1) Verify that the plug-in has been successfully loaded.
      2) Examine any debugging information externalized by the `toString()` method.
   c. Generate a scenario in which the function for the plug-in code should be performed and verify that it is working properly.
   d. If necessary, enable service tracing to help diagnosis problems. For more information on tracing, see “Tracing the Examples” on page 53.

Defining the Example Java Plug-Ins

From the Server

To run the example plug-ins while signed on to a server, do the following:

1. Go to the command prompt of a workstation on which the NetView management console server and console are both installed.
2. Change to the settings directory (where the `BINDIR` environment variable defines the installation root of the server): `cd %BINDIR%\TDS\server\db\current\settings`
3. Copy the following file to the %BINDIR%\TDS\server\db\current\settings directory: <console machine install path>\bin\generic_unix\TDS\client\examples\support\plugins.properties.

In Demo Mode

To run the example plug-ins in demo mode, from the console workstation, do the following:

1. Change to directory <install path>\bin\generic_unix\TDS\client\examples\support.
2. Issue the following command: `copy plugins.properties ..\..\settings`

Running the Example Java Plug-Ins

Running the Log Window Filter Plug-In: In a Live NetView Management Console System

Start the console by double clicking the standard desktop icon.

- To suppress message IHS2267:
  1. Select **Tasks → Send Message...** from the server or another console to broadcast a message to this console.

     The broadcast message entered is displayed (prefixed by IHS2267) in a message box. The example plug-in suppresses the addition of this message to the Log window.
- To suppress message IHS1107:
  1. Select **Tasks → Locate Resource** to open the Locate Resource window.
  2. Enter NEVER in the entry field and then click the **Locate** button.

     Message IHS1107 is displayed in a message box. The example plug-in suppresses the addition of this message to the **Log** window.

Note: Locate Resource generates message IHS1080 in demo mode.

Running the View Label Plug-In: In Demo Mode

To begin, start the console in demo mode by double clicking the **NMC Demo** desktop icon and do the following:

1. Select **Tasks → Locate Resource** to open the Locate Resource window.
2. Enter NTFFPU20 in the entry field and click the **Locate** button.
   - A view is displayed.
   - The names of the three nodes in this view are network qualified with USIBMNT.
   - The example plug-in suppresses the display of the network qualifier on the view (in either topology or details mode).
3. Position the cursor over the label of a node to display the fully qualified name in the fly-over section of the status bar.

Chapter 7. Configuring Property Files for Locally Launched Applications

NetView management console provides the capability to launch local applications on the NMC Console workstation, such as a web browser, to view a specific URL in context or to start a telnet session to the host where a managed resource resides. A properties file associated with these locally launched applications defines the specific executable file to launch for each supported platform. The Web browser and telnet commands are already configured. However, you can check the properties file to ensure that the commands specified can be successfully executed without a specific dependency on the current drive and directory. For all other local applications, define appropriate entries in the properties file.

Configure the local applications as follows:

1. Define which application executable file will be used on each platform where the topology console runs. This can be configured on a user basis or on a server-wide basis. This configuration is done at the topology server.
2. Define the command entries that will be added to the topology console pop-up menu. This configuration is done at the topology server.

Defining the Pop-up Menu Items

The cpebatch utility can be used to define commands that appear on the topology console pop-up menu. This is done by creating a response file and using it as input to the cpebatch utility. You can generate response files manually, with a standard text editor. To define an application to the NetView management console command menu, define the following variables in the command response file:

- `EXIT_NAME` to be `IHSXTJAM`
- `COMMAND_STRING` to be `com.tivoli.ihs.nmc.cmd.IhsCommandInvoker cmdname var1=value1 var2=value2...`

Although the command name is arbitrary, there should be a matching entry (`cmdname`) in the properties file for the command name to be resolved. If there is no match, then the command name is run as is.

Response File Input

To enable the application, command information must be added to the commands database through the command profile editor utility.

Following is an example entry of a command response file:

```
COMMAND = (
   NAME = CISCO_BLUE
   MENU_STRING = "CISCO APPN node detail view"
   RESOURCE_INDEP=NO
   HTML_HELP_FILE =
   HTML_HELP_ANCHOR =
   MIN_RESOURCES = 1
   MAX_RESOURCES = 1
   VERIFY = NO
   PAGE = (
      COMMAND_PLATFORM_LIST = GENERIC
      COMMAND_STRING = "com.tivoli.ihs.nmc.cmd.IhsCommandInvoker browser
URL=http://%RODM.ManagementURL%/cgibin/cw-blue/snamaps\?rqx=9&a;=nd&ip;=%ipaddress%&rc;=public"
      EXIT_NAME = IHSXTJAM
      MANAGER_NAME = ANY
```

```
        HTML_HELP_FILE =
        HTML_HELP_ANCHOR =
        CLIENT_PLATFORM_LIST = GENERIC
        TARGET_PLATFORM_LIST = GENERIC
    )
)
```

Note: This is only an example. Specify the http address for your own environment this section:

```
URL=http://%RODM.ManagementURL%/cgibin/cw-blue/snamaps\?rqx=9&a;=nd&ip;=
%ipaddress%&rc;=public
```

The entry for COMMAND_STRING determines how the command will be invoked. For example, the COMMAND_STRING for the `browser` command must start with the keyword `com.tivoli.ihs.nmc.cmd.IhsCommandInvoker` followed by `browser`.

There is a definition for the URL field in the COMMAND_STRING. A variable may be included in the URL field that will pull data from Resource Object Data Manager (RODM). Following is an example for the syntax of the variable name:

```
%RODM.ManagementURL%
```

This variable is not allowed on resource independent commands. For more detailed information on RODM variables, see Table 42 on page 64.

Creating a Response File for Browser

You can create response files manually, with a standard text editor. In addition, when creating a response file for the browser command, you can use the registration file conversion utility if a Distributed NetView registration file was provided by an equipment vendor. To manually create a response file, see “Response File Input” on page 61.

If you received a Distributed NetView Navigation Bar Registration file, you can use the registration file conversion utility to convert that file to a response file. Change to one of the following directories:

- For Intel: `cd %BINDIR%\..\generic_unix\TDS\client\bin`
- For UNIX: `cd $BINDIR/../generic_unix/TDS/client/bin`

Following is the syntax for the registration file `OEMNAVBAR.REG`:

```
tapp xx .. com.tivoli.ihs.nmc.server.IhsHttpParse <path>OEMNAVBAR.REG
```

Where *xx* is the appropriate platform from which the topology console is running. See “Appendix B. Topology Console Commands” on page 161 for more information about the **tapp***xx* command.

Note: The default output will be device dependent, but the menu entry will appear for any manager. Any changes to the response file need to be made before running the **cpebatch** command.

The previous syntax generates an `OEMNAVBAR.REG.rsp` file. The generated response file is used as input to the cpebatch utility as follows:

```
cpebatch OEMNAVBAR.REG.rsp -i -g
```

Changes to the right-click pop-up menu take effect immediately.

Defining the Properties File

The properties file is comprised of a heading section that defines generic information followed by sections defining a specific executable file and command string for each platform. Any line starting with a semicolon is considered a comment line. Comments cannot be on the same line as command text.

The following are file name types:

- Default File Name: %BINDIR%\TDS\server\db\current\settings\defaultcmdinv.properties
- User File Name: %BINDIR%\TDS\server\db\current\settings\<user name>cmdinv.properties

Note: The file name must be lowercase regardless of the user name.

In the properties file, define the generic and operating system sections. Table 40 displays the information for the generic section of the properties file.

Table 40. Generic Section of Properties File

Field	Description
command.desc	Defines the application entry. You can define as many different application specifications as necessary. This information is not used outside this file. In these examples, replace *command* with the name of the command you are defining.

Table 41 displays the information for the browser section of the properties file.

Table 41. Browser Section of Properties File

Field	Description
browser.usebuiltin	If set to TRUE, the built-in NetView management console browser is used. All operating system entries are ignored. This field is specific to the browser command and is not used for others.

Table 42 on page 64 displays the information for the operating system section of the properties file.

Table 42. Operating System Section of Properties File

Field	Description
command.x.platform	Identifies the operating system for which this entry is valid. Valid operating systems include the following: • OS/2 • Windows 95 • Windows 98 • Windows NT • Windows 2000 You can use wild cards (? or *) to specify multiple versions of an operating system. The file is processed from beginning to end until the first acceptable match is found. You can use the ? wild card to specify a one character wild card and the * wild card to specify a multiple-character wild card.

Table 42. Operating System Section of Properties File (continued)

Field	Description

command.x.run	Identifies the command used to run the executable file that will be run on this operating system. This executable file must be configured to run from the command line of the machine on which it will operate.
	
	For example, if a user's system has a Windows platform and the user must be in the c:\netscape directory to launch the web browser, their properties file might define the browser.x.run field as:
	
	`browser.1.run = c: & cd \\netscape & netscape`
	
	This example uses the (&) to string command line operations together, changing to the c:\netscape directory before issuing the netscape command. The & string concatenation technique is supported on Windows NT, Windows 2000, and OS/2 platforms only. You can use the batch file approach on other platforms which do not support stringing multiple command line operations together.
	
	Notes:
	
	1. Notice that you must specify a double back slash (\\) in the command line. In these properties files, the back slash is treated as an escape character similar to the way a C compiler treats a back slash. So when you need a back slash character in your path name, use two back slashes.
	2. Avoid using environment variables, as they might not be resolved in the final command string.
	3. You might want to specify a directory greater than eight characters in length, such as Program Files in the following example:
	
	`browser.1.run = cmd /c start /Dc:\\Program Files\\netscape\\netscape.exe`
	
	To be sure you run the preferred application, enclose your full path name in quotation marks, prefaced with a back slash, as in the following example:
	
	`browser.1.run = cmd /c start /D\"c:\\Program Files\\netscape\\netscape.exe\"`
	
	You can also create a script file in a known directory and specify the script file as the executable file. Then the contents of the script file can be written to change to the proper directory and start the browser.

Table 42. Operating System Section of Properties File (continued)

Field	Description
command.x.args	Identifies the argument that will be passed to the specified executable file. Anything enclosed in percent (%) signs is considered a substitution variable that can later be resolved by NetView management console or RODM, or passed in from the command response file. For example, the `%url%` substitution variable in the browser.x.arg field defines a specific URL to be displayed when the browser is started. The specific URL is defined in the command response file. The substitution variables are gathered from three locations: • in the command string keyword of the CPE response file, where the variable and value are defined as `var=value`. For example, see "Response File Input" on page 61 for the URL variable. • `%ihs.`*xxx*`%` substitution variables. The character strings (*xxx*) following the period are the keys in the `IhsCmdInfo.java` and `IhsResInfo.java` objects. The substitution variable is the value in these objects. See the NetView management console Productivity Kit for more information. • `%RODM.`*xxx*`%` prefix substitution variables. The character strings (*xxx*) following the period are the field names in RODM for the selected objects. These should be used only for resource specific commands.

The following example illustrates a properties file :

```
;-------------------------------------------------------------------------
; Your comments go here.
;-------------------------------------------------------------------------
 browser.desc       = Open Web browser and show URL
 browser.usebuiltin = <true | false>

 browser.1.platform = windows nt
 browser.1.run      = C:\\program files\\Plus!\\Microsoft Internet\\IEXPLORE.EXE
 browser.1.args     = %url%

 browser.2.platform = OS/2
 browser.2.run      = netscape.exe
 browser.2.args     = %url%

 browser.3.platform = windows 9?
 browser.3.run      = netscape.exe
 browser.3.args     = %url%

 browser.4.platform = aix *
 browser.4.run      = netscape
 browser.4.args     = %url%

 telnet.desc        = telnet to host

 telnet.1.platform = OS/2
 telnet.1.run = start /c /f telnet
 telnet.1.args = %d2cmdargs%

 telnet.2.platform = Windows *
 telnet.2.run = telnet
 telnet.2.args = %d2cmdargs%
```

```
telnet.3.platform = aix
telnet.3.run = xterm -e telnet
telnet.3.args = %d2cmdargs%

telnet.4.platform = Solaris
telnet.4.run = xterm -e telnet
telnet.4.args = %d2cmdargs%

telnet.5.platform = HP-UX
telnet.5.run = xterm -e telnet
telnet.5.args = %d2cmdargs%
```

Notes:

1. The telnet definitions are shipped as a default in the properties file. You might need to override these definitions to customize the command for your operating system.
2. The `%d2 cmdargs%` variable is needed to support the **RunData2** command. The **RunData2** command uses the Remote Console support in RODM that might define the telnet command to run. See "%REMOTECONSOLE%" on page 103 for more information. The `d2cmdargs` variable is assumed to contain an IP address.
3. The telnet support in NetView management console also defines `%d2cmdargs%` when launching a telnet session. The telnet command appears in the right-click pop-up menu in a view. The `d2cmdargs` variable is assumed to contain an IP address.

You can define as many different application specifications as necessary. Each one should be numbered incrementally. For example, the previous example shows five specifications for telnet, numbered incrementally from 1 to 5. There is no limit to the number of specifications you can create; however, no numbers can be skipped.

Part 3. Using NetView Management Console

Chapter 8. Operating the NetView Management Console

This chapter includes information about:

- Starting the topology server
- Starting the topology console
- Signing on from the topology console window
- Using the HOSTCMD command
- Stopping the topology server and topology console
- Retrieving inventory data on IP resources
- Writing server information to the NMC databases
- NetView features available through NMC

Starting the Topology Server

The following sections describe how to start the topology server either manually or automatically. This task is usually completed by a system administrator for all topology console operators. If you do not need to start the topology server, skip to "Starting the Topology Console" on page 73.

Note: If you receive messages about the topology server, see the topology console help index for information about the messages.

Starting the Topology Server from the Desktop Icon

For Windows, double-click the **Start NMC Server** icon from the desktop.

For OS/2, double-click the **Start NMC Topology Server** icon on the desktop to open the NMC Topology Server folder. In this folder, double-click the **Start the NMC Topology Server** icon.

Manually Starting the Topology Server

Perform the following steps to start the server and to activate the TCP/IP or LU 6.2 connection to the Tivoli NetView for OS/390 environment:

1. Open a workstation command window or shell prompt, depending on your operating system.
2. Change to one of the following directories:
   - For Intel: `%BINDIR%\TDS\server\bin`
   - For UNIX: `$BINDIR/TDS/server/bin`

   Note: BINDIR is an environment variable used by the Tivoli Framework to define path information used by Tivoli applications. For Windows and UNIX, if the Tivoli Framework is not installed on the workstation from which the topology server is run, the installation process will install the `setup_env.cmd` command file or `setup_env.sh` script file that will define `BINDIR`.

   For OS/2, the PATH environment variable has already been updated to contain `%BINDIR%\TDS\server\bin` for your convenience. For Windows and AIX, you might want to update the PATH environment variable to make executing topology server commands from any directory in a command prompt more convenient.

Depending on the path used during installation, the Framework for the topology server typically has one of the following BINDIR values:

- usr\local\Tivoli\bin*interp*
- local\Tivoli\bin*interp*
- Tivoli\bin*interp*, where *interp* is:
  - For Windows, w32-ix86
  - For OS/2, os2-ix86
  - For AIX, aix4-r1

3. Enter one of the following commands to start the topology server:
   - For Intel: tserver start
   - For UNIX: ./tserver start

   Notes:

   a. You must be the root user to start the topology server. If you use telnet to log in to a remote topology server machine and start the topology server, you must export the display to your local machine before entering this command.

   b. On a UNIX system, if the topology server has been manually stopped and cannot be restarted, see Step 4 on page 86.

Starting the Topology Server as a Windows Service

If the topology server is configured to start manually, select the **Control Panel → Services** control applet:

1. Select **Topology Communication Server** and click **Start**.
2. Select **Topology Server** and click **Start**.

If the topology server is configured to start automatically, it starts when the machine is started.

Starting the Topology Server as a Daemon

For UNIX, issue the following command to start the topology server processes at system startup and have them run as daemons:

```
$BINDIR/TDS/server/bin/config -d
```

Note: You must be the root user to issue this command.

Establishing Communication Between the NetView Host and the Topology Server

Use the NETCONV command to set up communications with the NetView host. The NETCONV command starts and stops an IP or LU 6.2 communication session between the NetView host and the topology server. This command is entered at the NetView command line.

- To activate a TCP/IP connection, enter the following:

  NETCONV ACTION=START IP=*topo_server_hostname*

  Where *topo_server_hostname* is the TCP/IP host name for the machine on which the topology server is running.
- To activate an LU 6.2 connection, enter the following:

  NETCONV ACTION=START LU=*topo_server_LU_name*

Where *topo_server_LU_name* is the 8-character LU name of the topology server that communicates with the NetView host to obtain information about the OS/390 environment.

For more information about the NETCONV command, refer to the *Tivoli NetView for OS/390 Command Reference.*

Note: This step is necessary regardless of how you start the topology server.

Starting the Topology Console

You can start the topology console by selecting a desktop icon in Windows and OS/2, or you can start it by issuing the command in line mode from any operating system, or from the Tivoli Desktop.

Selecting the Desktop Icon in Windows and OS/2

To start the Topology Console using Windows and OS/2, do one of the following:

- For Windows, an icon is created on the desktop. Double-click the icon to start the topology console.
- For OS/2, a folder named NMC Topology Console is created. After opening it, double-click **Start the NMC Topology Console**.

Using a Line Command

Change to the appropriate directory for your environment:

- For Intel: `\usr\local\Tivoli\bin\generic_unix\TDS\client\bin`
- For UNIX: `/usr/local/Tivoli/bin/generic_unix/TDS/client/bin`

From the directory, issue the appropriate operating system specific command to start the topology console:

- For Windows 95 and Windows 98: `tconsole95 .. -key nmc`
- For Windows NT and Windows 2000: `tconsoleNT .. -key nmc`
- For OS/2: `tconsoleOS2 .. -key nmc`
- For UNIX: `tconsole.sh .. -key nmc`

See “Appendix B. Topology Console Commands” on page 161 for complete information about all of the `tconsole` command line arguments and supported environment variables.

Note: To avoid problems when starting the topology console on UNIX systems, add the directory that contains the **xhost** command to the PATH environment variable.

Using the Tivoli Desktop Icon

To start the topology console from the Tivoli Desktop icon, do the following:

1. Double-click the **Topology Console** icon, or right-click the icon and select **Open** from the pop-up menu.

   The topology console Sign On window, as shown in Figure 18 on page 74 is displayed.
2. Complete the steps in “Using the Topology Console Sign On Window” on page 74.

To start the topology console in demo mode, do the following:

1. Right-click the **Topology Console** icon.
2. Select **Properties** from the pop-up menu.
3. Enter **-local** in the Command Line Arguments entry field.
4. Click **Change and Close**.
5. Do one of the following:
   - Double-click the **Topology Console** icon.
   - Right-click the **Topology Console** icon and Select **Open** from the pop-up menu.

   The topology console Sign On window, as shown in Figure 18, is displayed.
6. Complete the steps in “Using the Topology Console Sign On Window”.

Using the Topology Console Sign On Window

Figure 18 shows the topology console Sign On window.

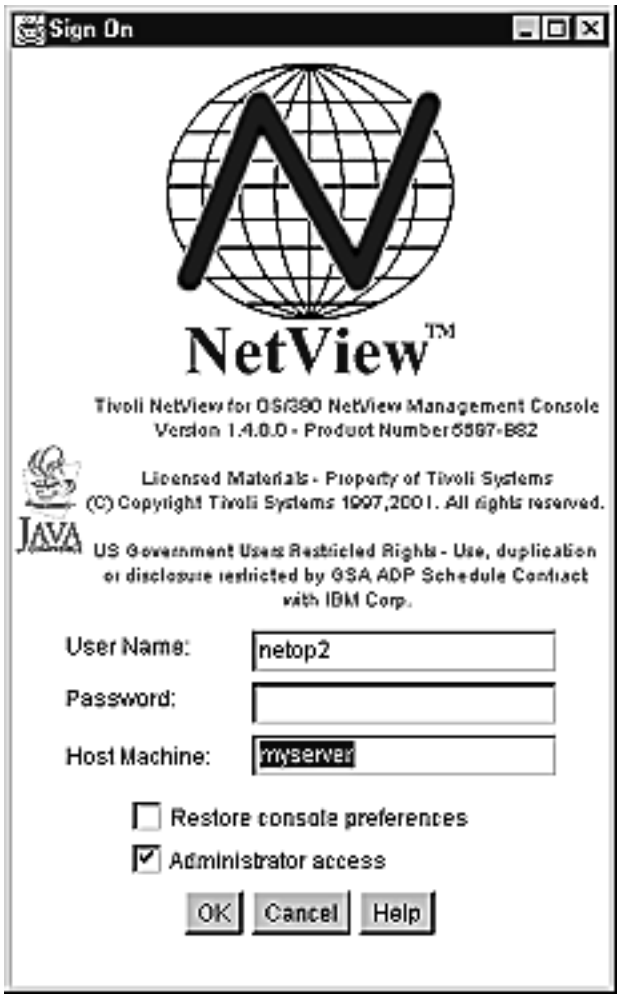


Figure 18. Sign On Window

The following steps describe how to sign on to the topology server.

1. Enter a user name for the topology server in the User Name entry field of the topology console Sign On window.

   The user ID must match your NetView user ID.
2. Enter a password for the topology server in the Password entry field.

   This password must match your NetView for OS/390 password.
3. Enter the IP host name of the topology server in the Host Machine field. If you reassign the topology server console port, specify the port number in this field the first time you sign on using this new port. However, the topology console retains the *server:port* setting on subsequent sign-ons and uses that setting as the default. See "Establishing Communication Between the NetView Host and the Topology Server" on page 72 for more information.
4. If you have previously signed onto NMC and saved the settings for the appearance of the topology console windows, you can select **Restore preferences**.
5. To use administrative functions, such as customizing settings and applying them to all topology consoles of this topology server, select **Administrator access**. More than one person can sign on with administrative access. In this case, one administrator can overwrite changes made by another administrator. The values saved last apply to all topology consoles.

   Note: For more information on authorization, see the "Defining Operators, Passwords, and Logon Attributes" chapter of the *Tivoli NetView for OS/390 Administration Reference* book. Specifically, see the `NGMFADMN` keyword.
6. Select **OK** to sign on to NMC. The topology console window, as shown in Figure 19 on page 77 is displayed.

   Note: The first time you start the NetView management console, expect a delay as support files are downloaded to the topology console. This will not happen on subsequent invocations of the topology console.
7. If a later level of the NetView management console is available, a message is displayed asking if you want to update the code.
   - If you choose **Yes**, the sign on window takes a short time while the latest level of code is downloaded. Then, the NetView management console is restarted and you can sign on again with the updated code.
   - If you choose **No**, sign on is terminated.

Notes:

1. If you want to run commands, ensure that the `NGMFCMDS` keyword is set to *yes* (`NGMFCMDS=yes`). The default is `yes`. For more information on the `NGMFCMDS` keyword, see the "Defining Operators, Passwords, and Logon Attributes" chapter of the *Tivoli NetView for OS/390 Administration Reference* book.
2. For more information about how files are automatically downloaded from the server at sign on, see "Appendix D. Automatic File Download at Console Log On" on page 169.

The Topology Console Window

Figure 19 on page 77 shows the topology console window. The following describes the areas on the topology console window:

- In addition to the menu bar, the tool bar contains selectable icons that provide a quick way to perform the most commonly used functions.

- The *business* view shows all views. Each node represents a view. You can click on the plus (+) or minus (-) signs beside a node to expand or collapse the node. Double-click on a node to open that view.

  Note: The icons in the business tree display actual status only in certain cases. The **Business Systems** branch of the business tree displays accurate status. The other branches of the business tree display a satisfactory status, though this is not an accurate representation of the status of the resources they contain. When a node that represents a resource is dynamically added to the business tree as a result of more-detail navigation, the status represents that of the resource. It does not represent the aggregate status of all resources in the view.
- The *animation* icon rotates when the topology console is communicating with the topology server. To cancel a request that is still in progress, click the icon.
- The work space contains the view area and view filter bar. See Table 11 on page 28 and “The Filter Bar” on page 78 for details about these two views on the work space. You can detach the work space to view multiple, different views at once.
- The log contains messages, issued commands, and command responses. Although command responses can be received asynchronously, they are always displayed following the issuing command.

  Note: The log is not displayed by default. To display the log, select **Options → Show Log** .

  When the visible portion of the log is full, the log begins to automatically scroll. To change this option so that you can manually scroll the log, right-click the log window and deselect **Automatic Scrolling**.
- The view bar displays buttons for all open views in the order they were opened. You can click any of the buttons in the bar to re-display a previously opened view.
- The status area displays the following information:
  - The text about the resource or business tree item under the mouse, or the status of any action in progress
  - The name of the topology server connection, your host name, the sign on time and date, the mode you are currently using, NETCONV status, and the topology console IP address
  - Information about the displayed view

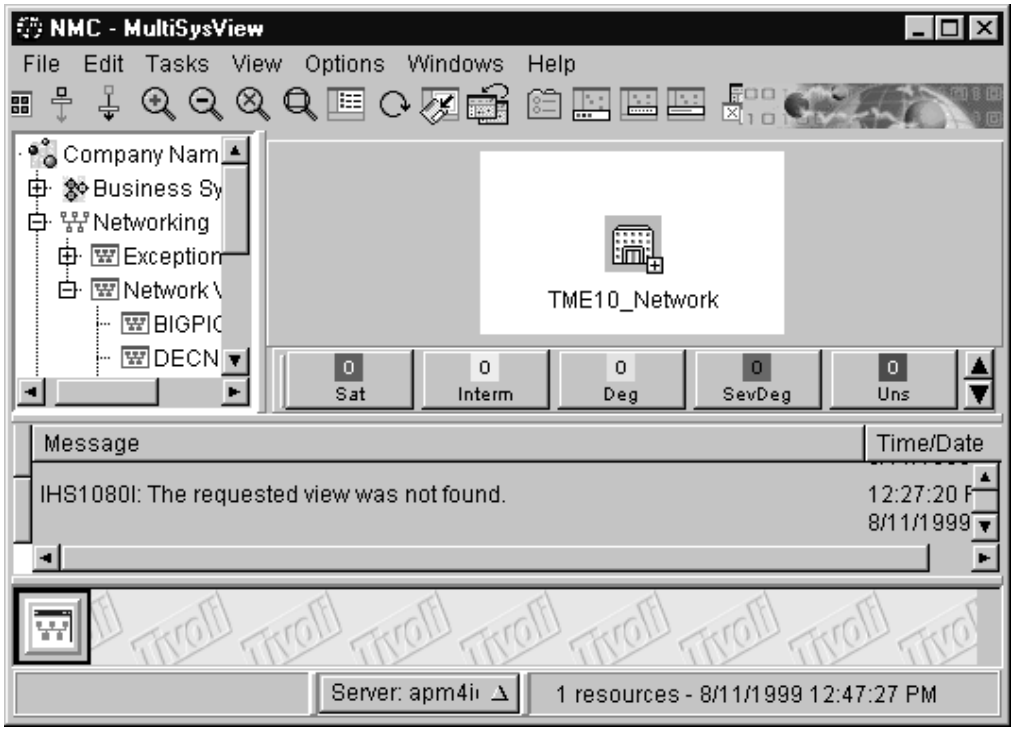


Figure 19. Topology Console Window

The View Area

The *view area* of the topology console window contains a notebook that displays different types of views of your resources. Resources can be displayed in topology views or details views.

- *Topology views* display resources and their statuses in a graphical format. Topology views are especially useful in displaying the relationships between resources; for example, displaying connectivity. Resources in a topology view are often placed on a background image, such as a building map or geographical map, which shows their general location and proximity to each other.
- *Details views* display resources and their statuses in a list format. Details views contain additional information about resources, such as type and description. They also enable you to control the order that resources are displayed, such as by time stamp or status. Null nodes and null links are never displayed.

Within a view, context menus contain various commands and functions available for a resource. To access a context menu, right-click a resource. A menu is displayed with a list of the available actions for that resource. To access a context menu for the view, move the mouse pointer to any area of the view that is not over a resource, then right-click. (If your details view is full, move the mouse pointer to any column heading, then right-click.) You will see a menu that displays the available actions for the view.

You can print a topology view or details view. Select **File** from the menu bar and then select **Print** from the pull-down menu to print a view. Note that you cannot zoom in on a topology view and print it. To print a larger picture of a view, re-size your window and then print.

The Filter Bar

The *filter bar* displays the number of resources in the current view at each status or by flag set. This status is indicated in views by the color of the resource.

You can use the filter bar to prevent resources with particular statuses from being displayed in the current view. To filter resources with a particular status, click on the corresponding button to that status on the filter bar.

- In topology views, filtered nodes are displayed as empty boxes and filtered links are displayed as dashed lines.
- In details views, filtered resources are not displayed.

The filter bar continues to display the number of resources in each status. To temporarily re-display a filtered resource in a topology view, including its status, icon, and so on, position your cursor on the resource without clicking.

You can customize the statuses that are displayed on the filter bar in the Status page of the Console Properties notebook.

NMC Online Help

NMC provides extensive online help from the topology console, which includes:

- Menu items.

  Note: To get help for menus, select **Help** from the tool bar and click **Menus help** in the menu.
- Dialogs and notebook pages.
- Windows, such as the Event Viewer window.
- Overview help.
- Messages for the topology server and topology console.
- A Legend function that displays a window containing descriptions of all resource types defined to the topology server.
- The ability to go outside the help facility to display a Web site.

You can also use the Help Index to search the help facility. A task index enables you to find help on specific tasks.

Notes:

1. If you receive messages about the topology server, look in the topology console Help Index for information about the messages.
2. For Command Profile Editor messages, see the CPE GUI online help.

NetView Management Console Features and Functions

The following sections provide a sampling of features and functions provided by the NetView management console.

Issuing UNIX/390 Commands from the NetView Management Console

The following UNIX/390 commands for IP can be issued from the NetView management console interfaces:

- Ping
- Tracert (trace route)
- Netstat (Network host status)

- Remote Ping

You can issue the available UNIX/390 commands from the NetView management console topology console screen in one of two ways:

- From a specific resource, right click on the resource and select **IP Commands from UNIX/390** from the context menu. This is available for any resource that contains an IP address.
- From the view background, right click the background and select **IP Commands from UNIX/390** from the pop-up menu.

From the **IP Commands from UNIX/390** menu, the following items are available:

- oping
- Remote Ping
- onetstat
- otracert
- MIB Browser
- Real Time Poller

The **oping** command is also available to be executed without opening a dialog when accessing this from a resource. When you select any of the other items, the IP address of the resource from which you accessed the menu is inserted into the Host machine field and you do not have to specify it. For more information on these dialogs and the options you can specify, see the NetView management console online help.

Issuing SNMP Commands from the NetView Management Console

The NetView program provides centralized management of TCP/IP and SNMP based nodes. From the NetView program, and from the NetView management console, full support is available for SNMP GET, SET, GETNEXT, GETBULK, WALK, and BULKWALK commands. These SNMP commands can be issued using OS/390 OSNMP and Java SNMP. The Java SNMP command communicates with:

- The OS/390 UNIX-based Java SNMP services to complete queries
- NetView for OS/390 for security

The Java Application Server

The Java Application Server (JAS) enables you to manage the following services in the UNIX/390 environment:

- SNMPSRVC
- POLLSRVC
- MIBPOLLING
- IP Discovery

JAS enables you to start, stop, and get status for each of these services. These services can still be started, individually, without starting JAS. Stopping JAS does not stop the processes that it is managing. For more information about starting JAS and the resources it manages, see the *Tivoli NetView for OS/390 Installation: Configuring Graphical Components* book.

The MIB Browser

The MIB Browser, from NMC, can interactively manage TCP/IP-based SNMP agents. The MIB browser communicates with:

- The OS/390 UNIX-based Java SNMP services to complete queries
- NetView for OS/390 for security

The MIB Browser supports SNMP GET and SET commands and displays table objects.

The MIB Browser also supports groups. You can expand the groups tree just as you do the MIB tree. There are 4 group items:

- List
- List+
- Walk
- Table

If you select List, Walk, or Table from the left frame of the MIB Browser, the right frame of the browser displays related information. If you click **List+**, the right frame of the browser displays limited information. To retrieve the SNMP data associated with this item you must fill in the Index field and click the **Apply** option button at the bottom of the screen.

Groups are defined in .grp files. Refer to the *Tivoli NetView for OS/390 Installation: Configuring Graphical Components* book for information about .grp files. A sample, /usr/lpp/netview/samples/fkxsnmp.grp file, is provided.

Figure 20 shows a MIB Browser.

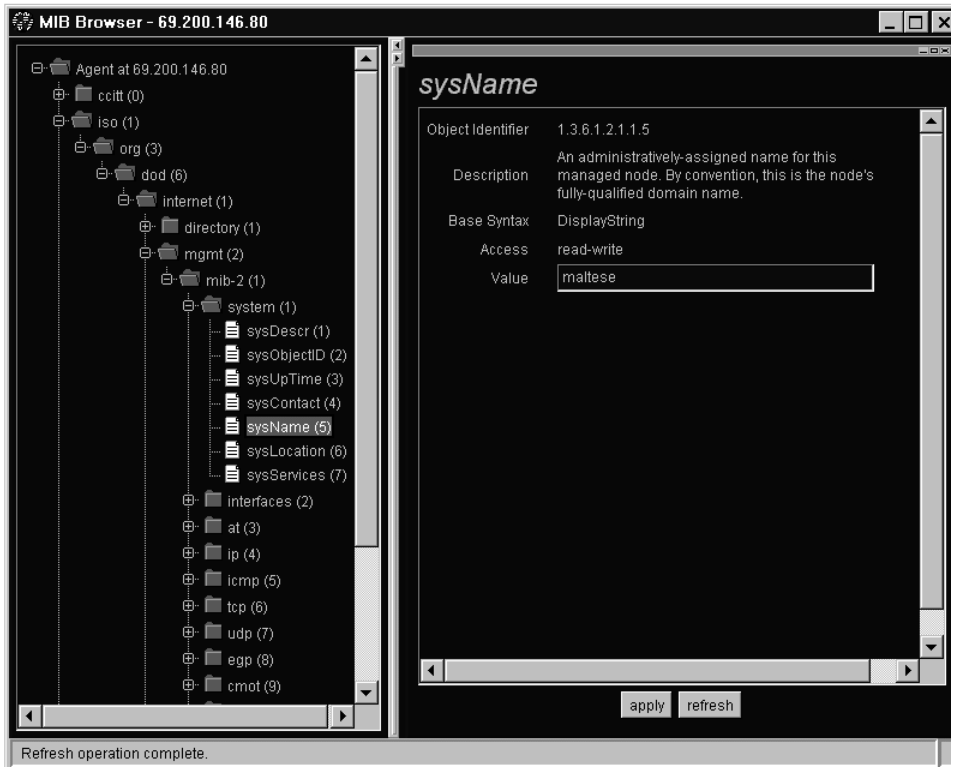


Figure 20. MIB Browser

Note: To view the online help for the MIB Browser, from the NetView management console main window, select **Help → Help Index** and then select **MIB Browser window** from the NetView management console index.

The Real-Time Poller

The Real Time Poller/Grapher enables you to view a real-time graph of the performance data of polling objects within hosts that you specify. Data for multiple polling objects, from multiple hosts, can be combined on a single graph. You can specify the polling interval for the polling objects and the maximum number of points that appear on the graph.

You can also perform functions such as changing the maximum points to display on the graph, adding new polling objects to the graph, starting and stopping the graph for individual polling objects, and selecting a new host. For more information, refer to the online help. Figure 21 shows an example of the Real-Time Poller with two MIB objects being polled.

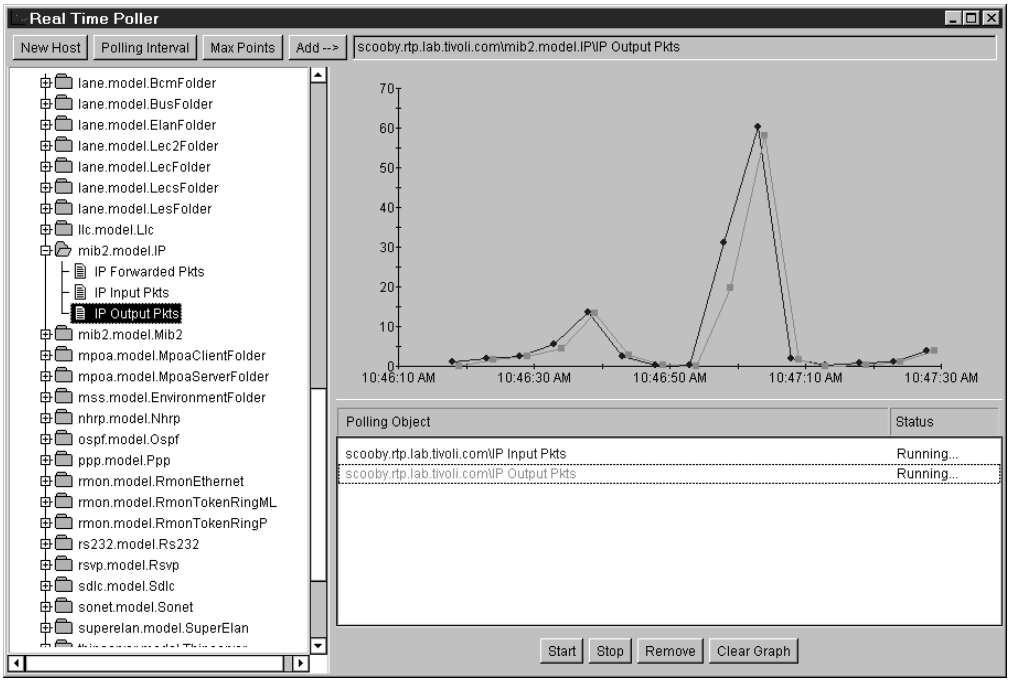


Figure 21. Real-Time Poller

Note: To view the online help for the Real Time Poller, from the NetView management console main window, select **Help → Help Index** and then select **Real Time Poller window** from the NetView management console index.

The NetView Resource Manager (NRM)

The NetView Resource Manager (NRM) enables you to manage all NetView programs in an enterprise with the NetView management console. The NRM collects task information and forwards this information to a *manager* NetView for processing. NetView Resource Managers require both Resource Object Data Model (RODM) and a NetView management console server to be operational. The processed information is stored in RODM.

With the NetView management console interface, NRM enables you to build views to monitor your enterprise. Once RODM is populated, you can use the NetView management console to monitor the NetView program. The views created by NRM are network views represented in the view tree as NetView Task views. Selecting a

NetView Task view results in a view of NetView programs. This view of NetView programs is the first NRM network object that can be selected from the NetView management console and it is an aggregate. Selecting **More Details** from this view opens a view containing aggregate objects which represent NetView tasks. Each NetView task aggregate can then contain real objects representing the following:

- CPU
- STG
- MQIN
- MQOUT
- MSGCT
- IO
- status (active/inactive)

Sample views are provided to monitor and manage NetView tasks.

If you want information about...	Refer to...
Sample views	*Tivoli NetView for OS/390 Resource Object Data Manager and GMFHS Programmer's Guide*
Using the NetView Resource Manager	*Tivoli NetView for OS/390 User's Guide* and *Tivoli NetView for OS/390 Installation: Getting Started*

Using the RODM Collection Manager With NetView Management Console

The RODM Collection Manager enables arbitrary grouping of objects into views or aggregates. Unlike BLDVIEWS which are static, the RODM Collection Manager dynamically manages the views or aggregates. This means that the RODM Collection Manager is continually updating the views or aggregates, enabling you to add, change, or delete collections. To access the RODM Collection Manager from the NetView management console, you must be signed on as an administrator.

Figure 22 on page 83 shows the RODM Collection Manager main menu.

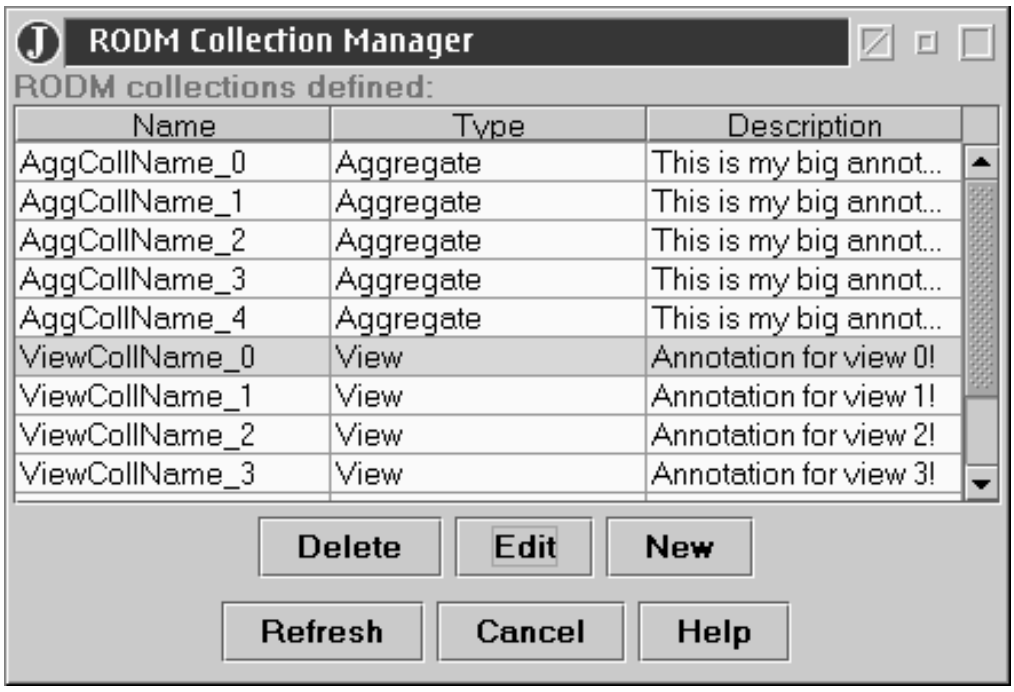


Figure 22. The RODM Collection Manager main menu

The NetView Inventory Server

The NetView Inventory Server enables a NetView management console operator to retrieve inventory information on a specific managed resource from a remote Tivoli inventory database. Inventory information can be obtained from any resource that has in IP address (in the Data1 field of RODM). Multiple types of queries can be performed, including user-defined queries.

Starting the NetView Inventory Server

To access the NetView Inventory Server from the managed node where NetAccess has been installed, perform the following steps:

For UNIX users:

1. Make the current directory `$BINDIR/NVInv`
2. Issue the following command: `./NVISStart.sh`

For NT users:

1. Make the current directory `%bindir%\nvinv`
2. Issue the following command: `bash nvisstart.sh`

Note: For this to work properly, NetAccess must be installed properly. To ensure that it is properly installed, issue the **winvgetdata** command from the command line. If this command is recognized, the Inventory Server has been installed properly and can be started. If the command is not recognized, see the *Tivoli Inventory User's Guide* to install the Tivoli Inventory Server.

The default port for this function is 8896. To change the port number, edit the NVISStart.sh and TivoliInventory.properties files. The NVISStart.sh file is located on the machine with the NetView Inventory Server, and the TivoliInventory.properties

file is located on the machine with the NetView management console server. These files are located in the following directories, respectively:

For UNIX users:

- `$BINDIR/NVInv`
- `$BINDIR/TDS/server/db/current/settings`

For NT users:

- `%bindir%\nvinv`
- `%bindir%\tds\server\db\current\settings`

Retrieving Inventory Data for Resources

You can potentially retrieve data from the NetView inventory server for any resource that has an IP address. To retrieve this data, use the **Get Inventory** command. To issue the command from the topology console, do the following:

1. Right-click a resource.
2. Select the **Get Inventory** command from the pop-up menu. If the chosen resource has information stored in the Tivoli Inventory Database, a second window opens, enabling you to select which query to run.

The command connects to the NetView Inventory Server located on a Tivoli managed node, which accesses the Tivoli inventory database. The information for the resource is returned to the topology console in a pop-up window.

Note: If the **Get Inventory** command does not process properly, click the **Help** button and follow the steps detailed in the help panel.

You can also create your own queries to retrieve specialized inventory data by configuring the TivoliInventory.properties file. For more information, see the "Installing and Configuring the NetView Management Console for Graphics" section of the *Tivoli NetView for OS/390 Installation: Configuring Graphical Components* book.

NetView Management Console Server Databases

Writing Server Information to the Topology Server Databases

Topology server databases are available in the directories shown in Table 43.

Table 43. Location of topology server databases

Database	Location
Default databases (Databases as installed)	For Intel: `%BINDIR%\TDS\server\db\default\datab` For UNIX: `$BINDIR/TDS/server/db/default/datab`
Current databases (Working set of databases)	For Intel: `%BINDIR%\TDS\server\db\current\datab` For UNIX: `$BINDIR/TDS/server/db/current/datab`
Backup databases (Backup of the current databases)	For Intel: `%BINDIR%\TDS\server\db\backup\datab` For UNIX: `$BINDIR/TDS/server/db/backup/datab`

To ensure that server information, such as views and resource information, is saved, manually initiate writing the information to the topology server databases. To manually write topology server information to a disk, issue the following command: `tserver utility -c`.

This command copies the view and resource information from the current databases to the backup databases and in-core information is written to the current databases. This should be done periodically so that view and resource information is saved to the disk in the event of an abnormal termination of the topology server. Otherwise, if the topology server terminates abnormally, the view and resource information might not be written to disk. See the server properties file for customization information which enables you to automatically backup the databases.

Note: If you do not manually write the databases to disk and the topology server terminates abnormally, after the next restart, the topology server will probably obtain the resource and view information from the source of this information.

When a view is customized and saved to the topology server, or, if the command profile editor saves commands to the topology server, they are written to the current databases immediately.

Note that view and resource information is also written to the databases when you shut down the topology server. During shutdown, the information in the current databases is copied to the backup databases prior to writing the in-core information to the current databases. View and resource information is also written to the databases when you shut down the topology server.

Note: You should regularly backup your topology server databases so that you can restore the databases if they become corrupted. For more information, see "Creating and Restoring a Permanent Copy of the Topology Server Databases".

When the Topology Server Databases Are Corrupted

If your topology server databases are corrupted, do the following:

1. Erase any files in the current and backup databases. Do not delete the directories.
2. Restart the topology server.

   This automatically copies the databases from the default databases to the current databases.

Creating and Restoring a Permanent Copy of the Topology Server Databases

To create a permanent copy of the server databases, perform the following steps:

1. Enter the following command:

   ```
   tserver utility -c
   ```
2. Copy the current databases into another directory of your choice. You might want to use a platform specific tool to compress these files into a single file.

To restore this permanent copy of the server databases, perform the following steps:

1. Stop the topology server. See "Stopping the Topology Server" on page 86 for more information.

2. Delete the contents of the current and backup databases. Do not delete the directories.
3. Copy the permanently stored databases saved in the previous steps into both the current and backup directories.
4. Start the topology server. See “Starting the Topology Server” on page 71 for more information.

Stopping the Topology Server

The procedure for stopping the topology server differs, depending on whether the topology server is running on an Intel or a UNIX platform. For Intel, you can stop the topology server by selecting an icon or entering a line mode command. For UNIX, you can stop the topology server only with a line mode command. The following sections describe the steps for stopping the topology server on each type of platform.

Attention: Do not stop the topology server by clicking the **X** in the top-right corner of the window.

With the Service Version on NT

To stop the topology server in this environment, select the **Control Panel → Services** control applet, or enter the **ihsxsrv stop** command from the `%BINDIR%\TDS\server\bin` directory on the topology server.

Using a Line Mode Command

To stop the topology server when it is running without problems, complete the following steps:

1. Open a workstation command window.
2. Change to one of the following directories:
   - For UNIX: `$BINDIR/TDS/server/bin`
   - For Intel: `$BINDIR%\TDS\server\bin`
3. Enter the **tserver stop** command to stop the topology server.

   The **tserver stop** command, with no flags, stops the topology server. For UNIX, only root users can stop the topology server.
4. For UNIX only, if the topology server is suspended, or if a daemon has trapped, issue the **tserver stop** command a second time with the `-f flag` as follows:

   ```
   tserver stop -f
   ```

   The **tserver stop -f** command forces running daemons to stop, and cleans up any remaining inter-process communication resources that were used by the daemons.

Stopping the Topology Console

To stop the topology console, select **File → Exit** from the topology console window. The topology console and all windows are closed.

Chapter 9. Using the NMC Command Profile Editor

When operators are using the topology console, available commands are displayed in the context menu for a selected resource. Also, resource independent commands are shown in a context menu if you right click on the view background. Because operators have different areas of responsibility, levels of authority, and preferences, you might want to customize the command menus. The command profile editor enables you to control the content, order, and capabilities of these menus for an individual operator or group of operators.

NMC provides a command profile editor GUI on Intel platforms and a batch utility on both Intel and UNIX platforms.

This chapter describes the command profile editor GUI. For more information about using the command profile editor, see the online help available from the command profile editor GUI. For information about using the command profile editor batch utility, see “Using the Command Profile Editor Batch Utility” on page 90.

Starting the Command Profile Editor

The command profile editor can be run only on an Intel platform that is supported by the topology server. To start the command profile editor, complete the following steps:

1. From the %BINDIR%\TDS\server\bin directory, enter: `cpe`

   The Sign On window is displayed.
2. Enter a valid user ID and password in the User Name and Password fields. The user ID and password must match your NetView for OS/390 user ID and password. Also, the user ID in this field must have administrative authority.
3. Select **Sign on**. A set of objects is displayed in the Command Profile Editor - Icon View window.

Note: The command profile editor does not support accessing a topology server on a different machine.

Understanding the Main Command Profile Editor Window

The first thing you see when you sign on to the command profile editor is the main Command Profile Editor window. This window is displayed as an icon view.

The Command Profile Editor window contains the following folder objects:

- Resource managers
- Commands and command sets
- Profiles
- Operators

Resource Manager Folder Objects

Resource manager objects represent applications that manage specific resources. The properties of a resource manager define a unique range of command indicators and specify the values in that range that are enabled for commands. You can use the command profile editor to add, change, or delete resource managers.

A command indicator is a numeric identifier assigned to a resource by its controlling resource manager. Each resource manager is assigned a unique range of values that it can define. Unless you are creating a resource management application or performing actions on behalf of such an application, you will probably never need to alter the properties of a resource manager.

Note: The first 32767 command indicators are available for customer use. Command indicators 32768 to 65534 are not defined and they are reserved for current and future Tivoli use. Refer to *Tivoli NetView for OS/390 Data Model Reference* for a detailed description of these command indicators.

Commands and Command Set Folder Objects

Command objects represent menu items that can be displayed in the context menus for selected resources. Command set objects also represent menu items, but command set items have submenus of commands or other command sets associated with them. The command set object specifies the order and content of the submenu. You can use the command profile editor to add, change, and delete commands and command sets.

Commands are composed of the following:

- A command definition page that defines the purpose and behavior of the command
- One or more implementation pages that describe the command exits that are responsible for carrying out the command

Command sets are composed of one or more commands or command sets.

You can find command and command set objects in the following windows:

- The Commands and Command Sets view
- The window of a profile object that contains the command or command set
- The window of a command set object that contains the command or command set

Regardless of where a particular icon is found, it represents the same object. Therefore, changes made to an object are seen by all users of that object.

Many commands have a common meaning for a wide variety of resource types. However, the mechanics of carrying out this command can vary from one resource type to another.

For example, a Lotus Notes® resource is started differently than a NetView resource, which is activated differently than a Tivoli Enterprise Console resource. This is because there is a variety of syntaxes and command exits; therefore command objects are composed of multiple implementation pages. When a command is issued, a single page is selected (based on the command indicator of the target resource).

For example, you can create a **Stop Tasks** command that is enabled for Lotus Notes resources, but not for Tivoli Enterprise Console resources. You could do this by defining a command called **Stop Tasks** and defining a page for Lotus Notes resources but not have a page for Tivoli Enterprise Console resources.

Profile Folder Objects

Profile objects represent a collection of commands. Through the commands, command sets, and separators, the profile objects define the content and order of the items on the context menus. You can use the command profile editor to add, change, and delete command profiles.

A profile can be shared by multiple operators. An operator object is created for each operator who does not want to use the default profile. The operator object is then assigned a profile object. Changes to a profile affect all operators using that profile. This method makes it easier to maintain profiles and ensures that operators are using the same commands.

Note: A profile named <default> is used for all operators who do not have a specific operator definition. The <default> profile is appended by the topology server during initialization when processing CDF/BDF files of instrumented applications. This is done for the Topology Display Subsystem view.

Operator Objects

An operator object represents a topology console operator and is associated with that operator by a common name. An operator object specifies the profile object assigned to the operator. You can use the command profile editor to add, change, or delete operators.

When an operator right-clicks a resource, the profile object used to populate the context menus for that operator is identified as follows:

- If an operator object exists with the same name, its specified profile object is used.
- If an operator object does not exist with the same name, but a profile object with the name <default> exists, the profile object is used.
- Otherwise, no commands from the command profile editor are displayed on the context menu.

Note: When you delete or rename the <default> profile, only operators with operator objects assigned to them will have access to commands from NMC. You can use this to provide security for a topology server.

Using the Command Profile Editor Window

From the main Command Profile Editor window, you can open one or more of the folders discussed in “Understanding the Main Command Profile Editor Window” on page 87. For more information, see “Opening Command Profile Objects” on page 90.

In addition to opening objects, you can perform the following functions from this window:

Save changes
To save the changes you have made with the command profile editor to the topology server, select **Save changes** from the **Profile editor** pull-down menu.

Arrange the view
To arrange the objects in the window, select **Arrange** from the **View** pull-down menu.

Display a detail view
To show details about the objects, instead of icons, select **Change View** and then select **Details** from the **View** pull-down menu. The main command profile editor window is displayed as a detail view.

The detail view displays additional information about the objects, including the date and time the objects were last modified and who modified them.

Select or deselect all objects in the view
To select or deselect all objects in the view, select **Select all** or **Deselect all** from the **Selected** pull-down menu.

Opening Command Profile Objects

To work with an object in the main command profile editor window, open a view of the object. You can open the following types of views of objects:

Icon view
Displays the items as icons

Tree view
Displays the items in a tree format

Detail view
Displays the items in a list format and includes details about them

Note: This chapter primarily uses icon views as examples. The functions available from each view type are the same.

To open an object, do one of the following:

- Double-click the object to open a view of the object.
- Right-click the object, then select one of the **Open as** options from the pop-up menu.
- Select the object, then select one of the **Open as** options from the **Selected** pull-down menu.

A view is opened for the selected object.

Stopping the Command Profile Editor

To stop the command profile editor, do the following:

1. Save the changes by selecting **Save changes** from the **Profile editor** pull-down menu of the main **command profile editor** window.
2. Close the main **command profile editor** window. All other command profile editor windows are automatically closed.

Using the Command Profile Editor Batch Utility

In addition to a user interface, the command profile editor provides a batch utility program that you can use to add, change, or delete commands in the command profile editor database. The batch utility runs on the Intel and UNIX platforms supported by the topology server. The utility uses a text file, called the response file that contains responses to all the parameters used in creating commands.

Note: NMC provides a utility to convert NetView Graphic Monitor Facility (NGMF) command tree facility definition files to command profile editor response files. See “Chapter 11. Converting NGMF Command Sets” on page 131 for more information.

The command profile editor batch utility can do the following:

- Create a response file from the information currently in the command profile editor database of the topology server. Note that the response file can be used as a backup for the command profile editor database or for synchronizing your topology servers.
- Update the command profile editor database of the topology server from a response file.

Any program can provide a response file and run the batch utility to add commands to the command profile editor database.

Two sample response files are provided with the command profile editor. These are available from the databases of the NetView management console Server. The samples are provided to enable you to customize the commands. The file names are:

ihsscpe
: Base commands that are shipped with the NetView management console Server

flccpe Commands for MultiSystem Manager (MSM) resources

These files are located as follows:

- `ihsscpe`, for Intel: `%BINDIR%\TDS\server\sample\ihsscpe.`*`xxx`*`.rsp`
- `ihsscpe`, for UNIX: `$BINDIR/TDS/server/sample/ihsscpe.`*`xxx`*`.rsp`
- `flccpe`, for Intel: `%BINDIR%\TDS\server\sample\flccpe.`*`xxx`*`.rsp`
- `flccpe`, for UNIX: `$BINDIR/TDS/server/sample/flccpe.`*`xxx`*`.rsp`

Note: *XXX* is a country code indicator, such as en_US.

Depending on the parameters specified on the command line, the utility adds, modifies, or deletes commands. The command profile editor utility does not save information to the topology server until the entire response file is processed and verified. If the entire file cannot be processed successfully, no updates are made in the topology server database.

Starting the Command Profile Batch Utility

To run the command profile editor batch utility, issue the **cpebatch** command from one of the following directories:

- For Intel: `%BINDIR%\TDS\server\bin`
- For UNIX: `$BINDIR/TDS/server/bin`

For more information about the **cpebatch** command, see “cpebatch” on page 144.

Input and Output Files of the Response File

The format for the input and output of the response file is identical. The response file is a standard ASCII file containing assignment statements that are generally in the form of *keyword = value*. The *keyword* is on the left side of the statement and identifies the parameter. The *value* is on the right side of the statement and either assigns a value to the keyword, or contains a block of keyword and value assignments.

Lines in which the first nonblank character in a line is an asterisk (*) are comments and are ignored by the utility. Comments can be either inside a block or outside a block, for example:

```
MANAGER = (
* This is a comment inside of the manager block
     NAME = SNA
     INDICATOR_HIGH = 10
     INDICATOR_LOW = 5
     INDICATOR.INDICATOR_LOW+0 = PU 2.1
     INDICATOR.INDICATOR_LOW+1 = PU 2.0
     INDICATOR.INDICATOR_LOW+2 = 3274
Communications Controller
     COMMENT = Defines manager SNA
)
* This is a comment outside of the manager block
```

The preceding example creates a manager named SNA and assigns it a range of command indicators from 5 to 10. It also creates three indicators for that manager, the lowest indicator defined is 5. For an example input file, see:

- For Intel: `%BINDIR%\TDS\server\Sample\ihsscpe.`*xxx*`.rsp`
- For UNIX: `$BINDIR/TDS/server/Sample/ihsscpe.`*xxx*`.rsp`

Note: *XXX* is a country code indicator, such as en_US.

The response file is processed in a single pass; therefore, the order of the main blocks of keywords is important. The order must be as follows:

1. Manager
2. Command
3. Command_Set
4. Profile
5. Operator

You cannot use items before they are defined in the response file. For example, if command set A embeds command set B, command set B must be defined first.

You can omit any block as long as all the required keywords are already defined in the topology server. For example, if you add a command and the resource manager is already defined in the topology server, it is not necessary to define the manager in the response file.

You can delete a block from the current database by placing the DELETE keyword inside the block.

Note: Before deleting a command or command set, it must first be removed from any profiles or command sets that contain it. To do this, use the `REMOVE_COMMAND` or `REMOVE_COMMAND_SET` option of the `MENU` keyword for each profile or command set that contains the command or command set.

You can delete the following types of blocks:

- Manager
- Command
- Command_Set
- Profile
- Operator

Manager Keywords

The manager block defines a manager and the command indicators that this manager controls. The keywords to define managers in command blocks are shown in Table 44.

Table 44. Manager command block keywords

Keyword	Explanation
Name	Uniquely defines the manager to the command profile editor.
INDICATOR_HIGH	High end of the range of indicators controlled by this manager.
INDICATOR_LOW	Low end of the range of indicators controlled by the manager.
INDICATOR_*n*	*n* must be the indicator value followed by the description string. The indicator value can be either a fixed number or can be relative to the INDICATOR_LOW+*x*, when *x* is the offset of this indicator from the low range.
COMMENT	Comment string. Limit of 256 characters.

Command Keywords

The command block defines a complete command, including general information and each of the command pages. The keywords used in command blocks are shown in Table 45.

Table 45. Command block keywords

Keyword	Explanation
NAME	Uniquely identifies the command to the command profile editor.
MENU_STRING	The string that is displayed on the context menu.
COMMENT	Comment string. Limit of 256 characters.
HTML_HELP_FILE	The name of the HTML file that contains the help for this command.
HTML_HELP_ANCHOR	The anchor, if applicable, to a particular location in the HTML file that contains the help for this command. The pound sign (#) in the first position is optional; when HTML is displayed by the NetView management console, # is inserted, if necessary.
MIN_RESOURCES	Minimum number of resources that must be selected in a view before this command is enabled. The range is 1 to 10.
MAX_RESOURCES	Maximum number of resources that can be selected in a view before this command is disabled. The range for this is *value of the minimum resources*–100, or an infinite number. An infinite number is denoted by the keyword INFINITY. **Note:** Specifying too many resources can cause system resources to become overburdened and, therefore cause the system to hang.
PAGE	Defines the characteristics of a page for an individual command. See “Page Keywords in the Command Block” on page 94 for the items you can specify here.

Table 45. Command block keywords (continued)

Keyword	Explanation
RESOURCE_INDEP	YES or NO. Indicates whether this command is enabled regardless of whether resources are selected. If the value of RESOURCE_INDEP is YES, then MIN_RESOURCES and MAX_RESOURCES values are ignored. Resource dependent commands are displayed when you right-click over a resource. Resource independent commands are displayed when you right-click over the view background.
VERIFY	YES or NO. Indicates whether to issue a confirmation message before the command is sent to the manager.

Page Keywords in the Command Block

The page keywords in the command block define the characteristics of a page for an individual command. The keywords to define pages in command blocks are shown in Table 46.

Table 46. Page command block keywords

Page Characteristic	Explanation

CLIENT_PLATFORM_LIST	The topology console platform or platforms associated with this command page.
	• The syntax for one platform is:
	`CLIENT_PLATFORM_LIST = platform`
	Where *platform* is the platform you are using. Valid platforms are: AIX, HP-UX, OS/2, Solaris, Linux, Windows 2000, Windows NT, Windows 98, and Windows 95.
	• The syntax for all platforms is:
	`CLIENT_PLATFORM_LIST = GENERIC`
	• The syntax for two or more platforms is:
	`CLIENT_PLATFORM_LIST = (`
	`CLIENT_PLATFORM = platform1,platform2`
	`CLIENT_PLATFORM = platform3,platform4`
	`)`
	Where *platform1*, *platform2*, *platform3*, *platform4*, and so on, are the platforms you are using.
	Note: When specifying more than one platform you must separate each platform with a comma.
	
	The topology console must be running on the specified platform to enable the command.
	Note: When running a response file in mode -m or -g, the list of platforms is added to the existing set of platforms. Use a platform of CLEAR to clear all platforms from the list. To clear all platforms and specify a new list of platforms, specify CLEAR, followed by a comma, followed by a specified list of platforms.

Table 46. Page command block keywords (continued)

Page Characteristic	Explanation

TARGET_PLATFORM_LIST	The target platform or platforms where the selected resource is located.
	• The syntax for one platform is:
	`TARGET_PLATFORM_LIST = platform`
	Where *platform* is the platform you are using. You can specify: unknown, Other, sunos4, solaris2, hpux9, hpux10, aix3-r2, aix4, winNT, win95, netware3, netware4, or NetView/390. Other platforms can be specified and will be added to the command profile editor list of valid platforms.
	• The syntax for all platforms is:
	`TARGET_PLATFORM_LIST = GENERIC`
	• The syntax for two or more platforms is:
	`TARGET_PLATFORM_LIST = (`
	`TARGET_PLATFORM = platform1,platform2`
	`TARGET_PLATFORM = platform3,platform4`
	`)`
	Where *platform1*, *platform2*, *platform3*, *platform4*, and so on, are the platforms you are using.
	Note: When specifying more than one platform, separate each platform with a comma.
	The target must be running on the specified platform to enable the command.
	Note: When running a response file in mode -m or -g, the list of platforms is added to the existing set of platforms. Use a platform of CLEAR to clear all platforms from the list. To clear all platforms and specify a new list of platforms, specify CLEAR, followed by a comma, followed by a specified list of platforms.

MANAGER_NAME	The name of the manager to which the page applies. This must be previously defined in the response file. If this command is not associated with particular resource types, specify the string `ANY`. An `ANY` manager command can have only one command page, which can not specify an INDICATOR_LIST or PAGE_ID.
PAGE_ID	Uniquely identifies this command page by specifying one indicator from the indicator list. When updating a command page, the PAGE_ID must be in the indicator list of the existing page in CPE.

Table 46. Page command block keywords (continued)

Page Characteristic	Explanation

INDICATOR_LIST	Defines the indicators that invoke this page. Specify single indicators by separating each with a comma. Specify a range by connecting numbers with a dash (for example, 1–100). You can replace real numbers with relative numbers using INDICATOR_LOW+*x*. INDICATOR_LOW denotes the lowest defined indicator for this manager.
	
	If the indicator list spans multiple lines, you can use the following format:
	`INDICATOR_LIST = (`
	`  VALUE.0 = INDICATOR_LOW + 0`
	`  VALUE.1 = INDICATOR_LOW + 1`
	`)`

COMMAND_LIST	The command string to be sent to the command exit specified by EXIT_NAME.
EXIT_NAME	The name of the command exit to invoke for this page. See “Using Topology Server Command Exits” on page 100 for information.
LU_NAME	Currently, this field is ignored by NMC. All exits are executed on the topology server workstation.
HTML_HELP_FILE	The name of the HTML file that contains the help for this command.
HTML_HELP_ANCHOR	The anchor, if applicable, to a particular location in the HTML file that contains the help for this command. The pound sign (#) in the first position is optional; when HTML is displayed by the topology console, # is inserted, if necessary.

Command Set Keywords

The command set block defines what a command set will look like. Order is important in the menu block. The keywords used in command set blocks are shown in Table 47.

Table 47. Command set block keywords

Keyword	Explanation
NAME	Uniquely identifies the command set to the command profile editor.
COMMENT	Comment string. Limit of 256 characters.
MENU_STRING	The string that is displayed on the **Context** menu.
HTML_HELP_FILE	The name of the HTML file that contains the help for this command set. This file must be installed on the topology server.
HTML_HELP_ANCHOR	The anchor, if applicable, to a particular location in the HTML file that contains the help for this command. The pound sign (#) in the first position is optional; when HTML is displayed by the topology console, # is inserted, if not already there.

Table 47. Command set block keywords (continued)

Keyword	Explanation
MENU	Defines the commands that this command set contains and their order. You can specify one or more of the following: • **COMMAND_NAME** - Specifies the name of a command to add. • **COMMAND_SET_NAME** - Specifies the name of a command set to add. • **SEPARATOR** - Specifies that a separator should be placed on the menu. • **REMOVE_COMMAND** - Specifies the name of a command to remove. • **REMOVE_COMMAND_SET** - Specifies the name of a command set to remove.

To add a new command to a command set or profile, first identify the command set or the profile to which you want to add the command, then specify the command you want to add inside the MENU block. The new command must already be defined in the database or must have been defined earlier in the response file.

For example, to add `My command` to the Network command set, specify the following:

```
COMMAND_SET = (
  NAME = Network               (identifies the
command set)
  MENU = (
    COMMAND_NAME = My command  (identifies
the command to add)
  )
)
```

This example places `My command` at the end of the Network command set. Make sure you use the `-G` option on the **cpebatch** command when adding the command.

If you want to insert `My command` after an existing command in the command set, specify the following:

```
COMMAND_SET = (
  NAME = Network
  MENU = (
    COMMAND_NAME = NetView command line   (existing
command)
    COMMAND_NAME = My command
  )
)
```

To remove `My command` from the Network command set, specify the following:

```
COMMAND_SET = (
  NAME = Network               (identifies the
command set)
  MENU = (
    REMOVE_COMMAND = My command  (identifies
the command to remove)
  )
)
```

Profile Keywords

The profile block defines individual profiles. Order is important in the menu block. The keywords used in the profile command block are shown in Table 48.

Table 48. Profile command block keywords

Keyword	Explanation
NAME	Uniquely identifies the profile to the command profile editor.
COMMENT	Comment string. Limit of 256 characters.
MENU	Defines the commands that this profile will contain and their order. You can specify one or more of the following: • **COMMAND_NAME** - Specifies the name of a command to add. • **COMMAND_SET_NAME** - Specifies the name of a command set to add. • **SEPARATOR** - Specifies that a separator should be placed on the menu. • **REMOVE_COMMAND** - Specifies the name of a command to remove. • **REMOVE_COMMAND_SET** - Specifies the name of a command set to remove.

Operator Keywords

The operator block defines operators to the command database. The keywords used in the operator command block are shown in Table 49.

Table 49. Operator command block keywords

Keyword	Explanation
NAME	Uniquely defines the operator to the command profile editor
COMMENT	Comment string. Limit of 256 characters
PROFILE_NAME	The name of the profile that this operator will use

Chapter 10. Using the Topology Server Command Exits

This chapter contains information about writing topology server command exits.

The command exit facility enables commands defined in the context menus for selected resources to invoke specific functions when those commands are selected. These specific functions are known as command exits and are executed as remote procedure calls. When a command is selected, its corresponding command exit procedure is driven by the topology server.

Command Profiles

Command profiles define the commands available from the context menus for a particular operator. When an operator right-clicks a resource or the view background, the command profile for that operator is used if it exists; otherwise, the default command profile is used.

A default command profile is shipped with NMC. You can add command definitions to the default command profile or modify existing definitions. You can create new command profiles for individual operators or groups of operators. Commands are also automatically defined by component instrumentation.

Understanding Topology Server Command Exits

The topology server provides a set of command exits to send commands from the workstation to the NetView host as described in “Using Topology Server Command Exits” on page 100. You can also write user command exits that get control when a command is selected as described in “Writing Topology Server Command Exits” on page 104. The command exit facility supports exit-to-exit communication, which enables a command exit to modify a command and to pass the data to another command exit for processing.

When a command exit returns, control is given to its invoker. Eventually, the first exit invoked by the command selection returns.

Note: There are some events that take place when a command exit is driven. If an exit procedure is not yet registered, then an executable file with the same file name as the exit name is sought using the defined path. If found, this executable file is started in a separate session.

For example, if the exit is called TESTEXIT, the topology server searches for and starts an executable program called TESTEXIT.EXE (for Intel platforms) or TESTEXIT (for UNIX platforms). It is the responsibility of this executable program to register a procedure within itself as a command exit procedure.

When a command is issued from the topology console, the command exit indicated in the command profile is driven by the topology server. For a resource dependent command, the command exit that is driven is based on the command indicator of the resource. A parameter block is passed to the command exit procedure containing information about the command in the command profile editor and information about a resource (if selected) when the command was invoked. The command exit procedure can pass a return code to its invoker when processing of the command is complete.

Using Topology Server Command Exits

The command exits that are supplied with the topology server for general use are shown in Table 50.

Table 50. Command exits supplied by the topology server

Command exit	Use to...	For information about...
IHSDGENE	Send a fixed set of generic commands to the NetView Graphic Monitor Facility host subsystem (GMFHS) for processing.	• Using with the command profile editor, see "IHSDGENE Command Exit".
IHSDNATV	Send commands to GMFHS, which forwards the command to the service point for the specified resource.	• Using with the command profile editor, see "IHSDNATV Command Exit" on page 101. • Invoking through a user-written command exit, see "IHSDNATV Command Exit" on page 123.
IHSXTHCE	Send a command to the NetView host.	• Using with the command profile editor, see "IHSXTHCE Command Exit" on page 101. • Invoking through a user-written command exit, see "IHSXTHCE Command Exit" on page 124.
IHSXTJAM	Launch a single Java class on the topology console for multiple resources.	• Using with the command profile editor, see "IHSXTJAM Command Exit" on page 102.
IHSXTJAV	Start a Java class on the topology console.	• Using with the command profile editor, see "IHSXTJAV Command Exit" on page 102. • Invoking through a user-written command exit, see "IHSXTJAV Command Exit" on page 126.

IHSDGENE Command Exit

For resources managed by GMFHS and MultiSystem Manager (MSM), use the IHSDGENE exit to send a fixed set of generic commands to NetView GMFHS for processing. Examples of the generic commands you can send include **Activate**, **Inactivate**, and **Recycle**. This exit supports only the commands listed in the default command profile and cannot be extended. To determine the commands supported by this exit and their syntax, look at the default command profile shipped with the topology server as defined in the ihsscpe.*xxx*.rsp and flccpe.*xxx*.rsp response files, where *xxx* is a country code indicator, such as *en_US*.

Note that you do not have to define the generic commands to the command profile editor as they are already defined in the response file.

For more information, refer to the *Tivoli NetView for OS/390 Resource Object Data Manager and GMFHS Programmer's Guide*.

IHSDNATV Command Exit

IHSDNATV is used to send a command to GMFHS, which forwards the command to the service point for the specified resource. A resource must be selected before IHSDNATV is invoked.

When defining a command that uses this exit, specify the following in the Commands notebook.

1. In the Command string field, specify the command to send to a network management gateway that manages the selected resource. GMFHS performs substitution for the following symbols in the command string:

   %appl%
   Substitutes the value of the TransactionProgram field of the Non_SNA_Domain_Class instance.

   %domain%
   Substitutes the value of the EMDomain field of the Non_SNA_Domain_Class instance.

   %resource%
   Substitutes the resource name portion of the MyName field of the GMFHS_Managed_Real_Objects_Class or a subclass of the GMFHS_Managed_Real_Objects_Class instance.

   For example, (`EMDomain.Resource = SPI6E69.MINI69A`) would cause `MINI69A` to be substituted.

   %spname%
   Substitutes the value of the MyName field of the `NMG_Class` instance.

   %type%
   Substitutes the value of the TypeName field of the Display_Resource_Type_Class instance associated with a resource.
2. In the Exit name field, enter `IHSDNATV`.

IHSXTHCE Command Exit

IHSXTHCE sends a command to the NetView host. If a resource is not selected when the command is invoked, IHSXTHCE can send resource-independent commands to the NetView host. If a resource is selected, the IHSXTHCE command exit can substitute resource specific information.

When defining a command that uses this exit, specify the following in the Commands notebook:

1. In the Command string field, specify the command to execute on NetView for OS/390. If you selected **Resource dependent**, you can optionally specify substitution variables in the Command string field. See “Substitution Variables” on page 102 for a list of valid substitution variables.
2. In the Exit name field, enter `IHSXTHCE`.

The following substitution variables are unique to the IHSXTHCE command exit.

%network%
Valid only for SNA topology manager Resource Object Data Manager (RODM) resources with command indicator values of 32769 and 32770. Substitutes the data before the first period in `display_name` in `EGVE_PARAMETERS32`, if it exists.

%noresponse%
Directs the command response back to the NetView for OS/390 session.

%resource%
Valid only for SNA topology manager RODM resources with command indicator values of 32769 and 32770. Substitutes the data after the last period in `display_name` in `EGVE_PARAMETERS32`.

%response%
Directs the command response back to the topology console.

IHSXTJAM Command Exit

The IHSXTJAM command exit starts a Java class on the topology console. The Java class name to be started must be the first blank delimited token in the command string field. Unlike the IHSXTJAV command exit, the IHSXTJAM command exit launches only one instance of the Java class when multiple resources are selected on the topology console. Thus, one instance of the Java class will have access to information on every selected resource. The IHSXTJAM command exit can be specified in a resource independent or a resource dependent command. See “Chapter 6. Topology Console Java Applications and Plug-ins” on page 51 for information about installing this Java class.

Note: The IHSXTJAM command exit cannot be called using the IhsiSend API as described in “IhsiSend - Remotely Invoke a Command Exit” on page 118, because this API can only send information about one resource or be resource independent. Use the IHSXTJAV command exit instead.

IHSXTJAV Command Exit

The IHSXTJAV command exit starts a Java class on the topology console. The Java class name to be started must be the first blank delimited token in the command string field. Unlike the IHSXTJAM command exit, the IHSXTJAV command exit launches multiple instances of the Java class, when multiple resources are selected on the topology console. For example, when two resources are selected, and a command is selected which invokes the IHSXTJAV command exit, two instances of the class specified in the command string launched are on the topology console, with each instance of the Java class having information about one of the selected resources. The IHSXTJAV command exit can be specified in a resource independent or a resource dependent command. See “Chapter 6. Topology Console Java Applications and Plug-ins” on page 51 for information about installing this Java class.

Substitution Variables

Table 51 lists the common variables that are substituted by the topology server in the command string for the command exits. Note that the substitution variables are not case-sensitive. These variables are common across all command exits.

Table 51. Common substitution variables

Substitution Variable	Description
%data1%	Substitutes the data1 from EGVE_PARAMETERS32.
%data2%	Substitutes the data2 from EGVE_PARAMETERS32.
%data3%	Substitutes the data3 from EGVE_PARAMETERS32.
%data4%	Substitutes the data4 from EGVE_PARAMETERS32.
%hb_hostname%	Substitutes the hb_hostname from EGVE_PARAMETERS32. Valid only for instrumented resources.

Table 51. Common substitution variables (continued)

Substitution Variable	Description
%hb_origin%	Substitutes the hb_origin from EGVE_PARAMETERS32. Valid only for instrumented resources.
%hb_primary%	First key value pair of hb_origin field from EGVE_PARAMETERS32. Valid only for instrumented resources.
%hb_secondary%	First key value pair of hb_sub_origin field from EGVE_PARAMETERS32. Valid only for instrumented resources.
%hb_source%	Substitutes the hb_source from EGVE_PARAMETERS32. Valid only for instrumented resources.
%hb_sub_origin%	Substitutes the hb_sub_origin from EGVE_PARAMETERS32. Valid only for instrumented resources.
%hb_sub_source%	Substitutes the hb_sub_source from EGVE_PARAMETERS32. Valid only for instrumented resources.
%ipaddress%	Substitutions data3 from EGVE_PARAMETERS32. Valid only for RODM resources.
%label%	Substitutes the display_name from EGVE_PARAMETERS32.
%monitor%	Substitutes the monitor name from the topology server database. Valid only for instrumented resources.
%objectid%	Substitutes resource_RODM_id in EGVE_PARAMETERS32.
%remoteconsole%	See “%REMOTECONSOLE%” for more information.
%tme_oid%	Substitutes the resource_TME_oid from EGVE_PARAMETERS32. Valid only for instrumented resources.

%REMOTECONSOLE%

The `%REMOTECONSOLE%` command line substitution variable applies only to RODM resources. The value for `%remoteconsole%` comes from the data2 field in the `EGVE_PARAMETERS32` structure, which comes from the DisplayResourceUserData field in RODM. The purpose of this substitution variable is to invoke a command or application on the topology console workstation.

For an example of how this substitution variable works with the Command Profile Editor and the topology console, see the command definition for Run Data2 Command in the `%BINDIR%\TDS\server\sample\ihsscpe.`*xxx*`.rsp`, where *xxx* is the country code, such as en_US. If syntax `RemoteConsole=/(can be anything)/` exists in the data2 field, then `%REMOTECONSOLE%` is substituted using the following rules:

- Syntax: `"RemoteConsole = /the_command the_args(0-n)/"`
- Fixed portions of this syntax are not case sensitive (such as RemoteConsole).
- Spaces around the equals sign are optional.
- Spaces between the first delimiter and the `the_command` are optional.
- The delimiter / can be any character. The first nonblank after the equals becomes the delimiter.
- A second occurrence of the delimiter character must exist after the first occurrence.
- There must be a nonblank character between the delimiters.

- The command is assumed to be a valid command on any topology console workstation that executes this menu item. To map a command to an appropriate command for the platform of the topology console workstation, update the usercmdinv.properties file on the Console. See “Chapter 7. Configuring Property Files for Locally Launched Applications” on page 61 for more information.
- This syntax can occur anywhere inside the data2 field. That is, there can be other characters before or after this syntax.

The %REMOTECONSOLE% substitution variable will be converted to the following:

```
the_command d2cmdargs="the_arg1 the_arg2 the_arg3"
```

The fixed characters are `d2cmdargs="` and the second double quote after the last `arg`. All other values are obtained from between the two delimiters.

Writing Topology Server Command Exits

The command exits that are supplied with the topology server which can be invoked through user-written commands are shown in Table 52.

Table 52. Command exits that can be invoked by user-written commands

Command exit	Use to...	For information about...
IHSDNATV	Send a command to GMFHS, which forwards the command to the service point for the specified resource. It can be specified as the command exit name when defining a command profile and can be invoked using the C programming interface.	• Invoking through a user-written command exit, see “IHSDNATV Command Exit” on page 123 • Using with the command profile editor, see “IHSDNATV Command Exit” on page 101
IHSXTHCE	Send a command to the NetView host. It can be specified as the command exit name when defining a command profile and can be invoked using the C programming interface.	• Invoking through a user-written command exit, see “IHSXTHCE Command Exit” on page 124 • Using with the command profile editor, see “IHSXTHCE Command Exit” on page 101
IHSXTJAV	Start a Java class on the topology console. It can be specified as the command exit name when defining a command profile and can be invoked using the C programming interface.	• Invoking through a user-written command exit, see “IHSXTJAV Command Exit” on page 126 • Using with the command profile editor, see “IHSXTJAV Command Exit” on page 102
IHSXTJCR	Send command responses to the topology console for display in the console log area. It can be invoked using the C programming interface.	• Invoking through a user-written command exit, see “IHSXTJCR Command Exit” on page 127

Available Programming Languages

You can write command exits in the C programming language. C exits offer the full range of the command exit functions; for example, a C exit can directly invoke another command exit (provided with the topology server or user-written). The VisualAge® tool set must be used.

Command Exit Flow Scenario

The following topics describe the sequence of events when invoking command exits for NetView commands.

1. A resource is selected in a view and a command is initiated by selecting a command name from a context menu. A command request is sent to the topology server.
2. The command request is processed by the topology server, which invokes the command exit procedure specified for this command. The user exit is invoked as a remote procedure call and is presented with information about the command (for example, the command string) and the selected resource.
3. The user exit performs processing that invokes the IHSXTHCE command exit to send a command to the NetView host.
4. An execute command request is sent to the appropriate NetView domain.
5. A response to the execute command request is received. This response indicates that the command was received by the NetView host, but it does not return the command responses.
6. The IHSXTHCE exit returns control to the user exit.
7. The user exit completes, returning a return code to the topology server that invoked the user exit.
8. The topology server returns this return code and the command string to the topology console, which displays error messages if necessary. The command string and any error is reported to the topology console log.
9. The command responses are sent to the topology server.
10. The command responses are sent to the topology console, and are displayed on the topology console log.

Command Exit for NetView Commands

Figure 23 on page 106 illustrates the sequence for invoking a typical command exit for NetView commands:

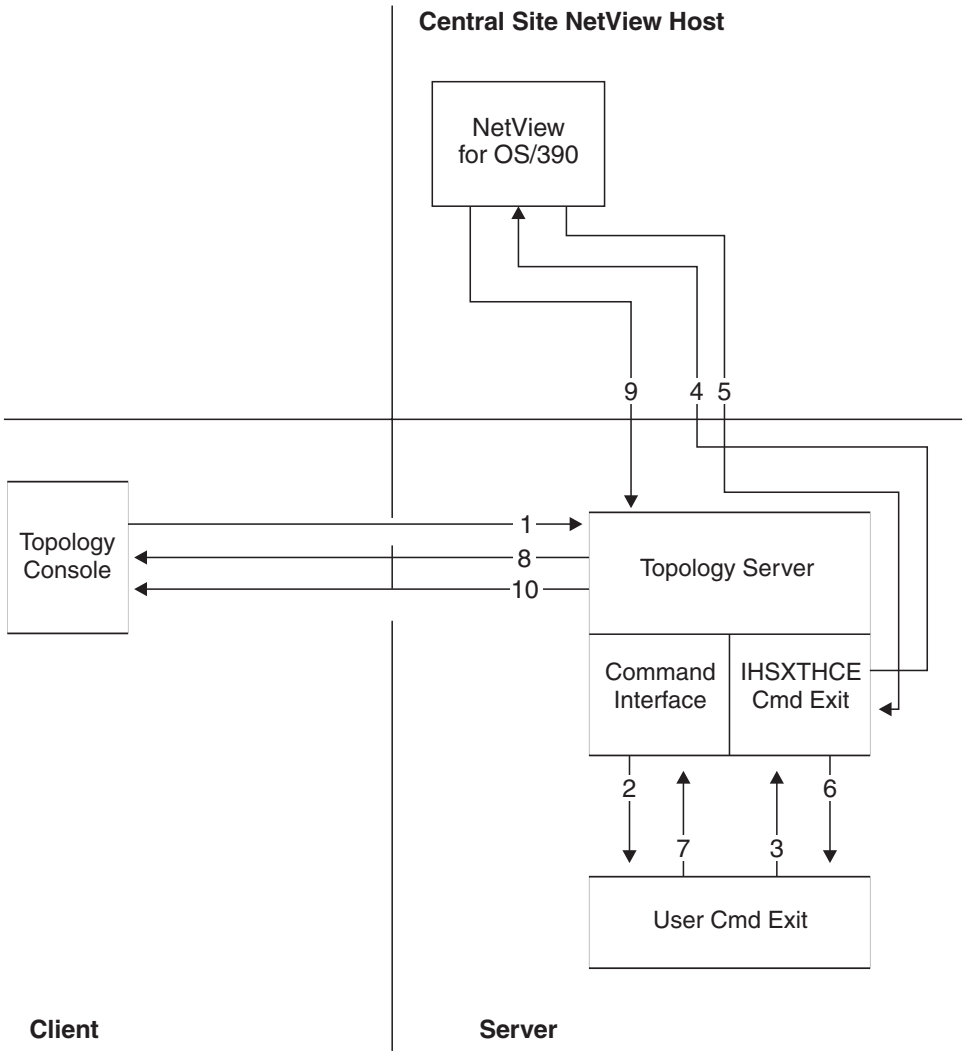


Figure 23. Components and Events Involved in Command Exit Invocation

Writing C Exits

This section describes how to write command exits in C. It also discusses the appropriate interfaces and language-dependent restrictions. The C language interface for command exits only supports 32-bit programs.

Building C Command Exit Programs

When building programs, use an option that instructs the compiler to use the multi-threaded version of the C library. The C source file using the command exit interface functions should include the ihsiexit.h header file. This file defines the relevant command exit data structures and function prototypes and is part of the topology server. The ihsiexit.lib library must also be specified when linking.

The ihsiexit.h header file is installed in one of the following:

- For Intel: `%BINDIR%\TDS\server\sample` directory
- For UNIX: `$BINDIR/TDS/server/sample` directory

The ihsiexit.lib file is installed in one of the following:

- For Intel: `%BINDIR%\TDS\server\lib` directory
- For UNIX: `$BINDIR/TDS/server/lib` directory

The topology server provides the exit ihssamce.c sample command in the following directories:

- For Intel: `%BINDIR%\TDS\server\sample`
- For UNIX: `$BINDIR/TDS/server/sample`

Following is an example of the absolute minimum that is needed for a command exit:

```
unsigned int exit_function_name(EGVE_PARAMETERS32_PTR pParms)
{
/* Put your logic here */
}
int main( int    argc
        , char **argv )
{
  IhsiInitialize( "EXIT"
          , exit_function_name );

  IhsiWait( "EXIT"
          , -1 );
  IhsiTerminate( "EXIT" );
  return;
}
```

Installing C Programs as Command Exits

A C program must be installed before it can be invoked as a command exit. If the name of the executable file matches the exit name, the topology server starts the executable file on the topology server workstation the first time the command is selected. Otherwise, it is the user's responsibility to start the executable file on the topology server workstation before the command is selected. To have the topology server automatically start the executable file, make the file name of the executable file the same as the exit name, and use an .exe extension if you are working on an Intel platform.

To define a C program (for example, CSAMP) as a command exit:

1. Use the command profile editor to define a command profile entry for CSAMP, specifying **CSAMP** as the exit name and add the command to the appropriate profile. For more information on using the command profile editor, see “Chapter 9. Using the NMC Command Profile Editor” on page 87
2. Use the application programming interface, described in “Command Exit Functions for C” on page 117, and a C compiler, to develop a program called CSAMP.
3. Ensure that the directory that contains CSAMP is included in the path, or copy CSAMP to the bin directory of the topology server.

When you select **CSAMP** from a context menu, the following events occur:

1. The topology server checks to see if CSAMP is already registered as an exit and attempts to start CSAMP if CSAMP is not registered.
2. The exit procedure in CSAMP is invoked with the information described in “C Parameter Block (EGVE_PARAMETERS32 Structure)” on page 108.

The C exit procedure keeps operator interaction to a minimum and returns immediately to the command exit handler when it is finished processing. Avoid continuous loops because they can tie up the command exit handler and use up system resources.

C Parameter Block (EGVE_PARAMETERS32 Structure)

When a C command exit is invoked, a pointer is passed to a parameter block of type EGVE_PARAMETERS32, which is a structure containing exit-specific information. The EGVE_PARAMETERS32 structure is defined in the ihsiexit.h C include file located in one of the following:

- For Intel: `%BINDIR%\TDS\server\sample`
- For UNIX: `$BINDIR/TDS/server/sample`

A user-written command exit also passes a pointer to a EGVE_PARAMETERS32 structure when invoking another exit from a C program. Table 53 describes the EGVE_PARAMETERS32 fields.

Table 53. The EGVE_PARAMETERS32 fields

Field	Description	Type (Size)
business_sys_name	Resource business system name.	PSZ (4 bytes)
client_handle	The IBM handle of the topology console that issued the command.	ULONG (4 bytes)
client_hostname	The TCP hostname of the topology console that is invoking the command.	PSZ (4 bytes)
client_ip_addr	The address of the topology console invoking this command. This is an IBM internal version of the socket file descriptor associated with the server's IP connection with the topology console.	SZ (9 bytes)
client_ip_addr_dd	IP dotted decimal address of the topology console that is invoking the command.	PSZ (4 bytes)
command_indicator	This field determines the command page used in command profile editor to execute this command.	USHORT (2 bytes)
command_name	The name of the command. If the exit is invoked from a topology console, this is the menu text from the command profile editor without any mnemonics.	PSZ (4 bytes)
command_string	The command string.	PSZ (4 bytes)

Table 53. The EGVE_PARAMETERS32 fields (continued)

Field	Description	Type (Size)
correlation_id	A unique number used to correlate requests and responses. This field in the EGVE_PARAMETERS32 structure should not be modified by a user command exit. Altering this field could result in an error when this exit returns to its invoker. This number is used by the topology server to correlate the command to its command responses. The command string is sent by the topology server to the topology console if you set want_corr_of_cmd_rsps before returning from your registered exit procedure. Then you can send command responses (using the IhsiSend API) to your command by calling the IHSXTJCR command exit using the same correlation number. If your exit is calling another exit and you create a new instance of the EGVE_PARAMETER32 block, copy the correlation_id passed from the caller into the new instance.	struct EGVE_CORR_ID (4 bytes)
data1	Resource specific data1. For RODM resources, this is data from the DisplayResourceOtherData field. For RODM predefined views in the tree view, this data is from the Annotation field.	PSZ (4 bytes)
data2	Resource specific data2. For RODM resources, this data is from the DisplayResourceUserData field.	PSZ (4 bytes)
data3	Resource specific data3. For RODM resources, this data is from the iPAddress field.	PSZ (4 bytes)
data4	Resource specific data4.	PSZ (4 bytes)
data_source	This field indicates the source of the resource. The possible values are: • **0** - All RODM resources, except for SNA Shadow objects and SNA Topology Manager resources • **1** - RODM SNA Shadow objects (from the GMFHS_Shadow_Objects_Class) • **2** - Topology Display Manager resources (Business System resources managed by the topology server) • **5** - RODM SNA Topology Manager resources	USHORT (2 bytes)
display_name	The resource name assigned by the manager of the resource. This is the name that appears in the view. For RODM resources, this data is from the DisplayResourceName field.	PSZ (4 bytes)

Table 53. The EGVE_PARAMETERS32 fields (continued)

Field	Description	Type (Size)
display_resource_type	The resource type assigned by the manager of the resource. This string appears in the: • Resource Properties dialog • In the Type column of a view in details mode	PSZ (4 bytes)
exit_executable_name	The name of the executable file for the invoked exit. This is optional and is used only when the name of the exit, as specified in exit_name, is not the name of the executable file to be run. Nulls or blanks indicate that it is not specified.	SZ (9 bytes)
exit_name	The name of the destination exit. This field is a null-terminated string padded on the right with blanks. This field structure should not be modified by a user command exit. Altering this field could result in an error when this exit returns to its invoker.	SZ (9 bytes)
exit_timeout	Used by some exits to timeout, in seconds, extended operations. • 0 indicates that a timeout has not been specified. • -1 specifies to wait indefinitely. • -2 means do not wait, and ignore the command responses.	LONG (4 bytes)
hb_es_managed_node	An enum value pair list used to identify the managed node names of various Tivoli Enterprise Console event servers. The topology server can use this to access the Tivoli Enterprise Console program and gather information. The format for the *hb_es_managed_node* slot is: • 1=<*The managed node name of the event server to which instrumentation sent a heartbeat*> This is not filled in for NetView instrumentation. • 2=<*The managed node name of the event server by which ihstetec was driven*> This is always filled in. An example of this field is: `1=kid.gt.com;2=pup.dg.org` Does not apply to RODM resources. Only available for instrumented resources.	PSZ (4 bytes)

Table 53. The EGVE_PARAMETERS32 fields (continued)

Field	Description	Type (Size)
hb_hostname	In the distributed environment, the value must be the TCP/IP host name of the system. In the OS/390 environment, it is the NetView domain name (non-network qualified). Each of these is used to direct the execution of commands to the correct system: distributed using Tivoli tasks or host using NetView command support. Does not apply to RODM resources. Only available for instrumented resources.	PSZ (4 bytes)
hb_origin	An enum value pair list used to identify the system on which the component resides. The enumerations are listed in the AMS document in relation to the Discover Connections task in the Connection Type group (Primary). Does not apply to RODM resources. Only available for instrumented resources.	PSZ (4 bytes)
hb_source	Contains a semicolon delimited triplet with information from the ComponentID group in the applications global description file (GDF) (manufacturer;product;version). Does not apply to RODM resources. Only available for instrumented resources.	PSZ (4 bytes)
hb_sub_origin	An enum value pair list used to differentiate between components on the same system that support the same relationship. Also described with discover connections (secondary). Does not apply to RODM resources. Only available for instrumented resources.	PSZ (4 bytes)
hb_sub_source	Same format and attribute use as source, but from the component description file (CDF). The topology server uses this field to locate the correct CDF. Does not apply to RODM resources. Only available for instrumented resources.	PSZ (4 bytes)
IBM_data	Tivoli use only.	PBYTE (4 bytes)
IBM_data_length	The length of the Tivoli use only data.	USHORT (2 bytes)
IBM_menu_id	The Tivoli internal command profile editor menu ID.	ULONG (4 bytes)
nv390_domain	NetView domain ID. When invoking a command exit from the context menu, the topology server primes the value with the domain ID of the NetView host to which the topology server is connected.	SZ (6 bytes)
nv390_hostname	TCP hostname of the NetView host, if available.	PSZ (4 bytes)
nv390_ip_addr	IP dotted decimal address of the NetView host, if available.	PSZ (4 bytes)
op_id	The NetView operator ID, if available.	SZ (9 bytes)

Table 53. The EGVE_PARAMETERS32 fields (continued)

Field	Description	Type (Size)
res_IBM_id	The topology server ID, for the resource. This field contains a zero for a resource independent command.	ULONG (4 bytes)
resource_flags	The flags of the selected resource. Zero, one, or more values can be set at one time. Possible values are: **0x80000000** Marked **0x40000000** Suspended **0x20000000** Automatically clear suspended **0x10000000** SNA alert pending **0x08000000** Threshold inconsistency **0x04000000** Automation in progress **0x02000000** Not monitored **0x01000000** IBM reserved 1 **0x00800000** Child suspended **0x00400000** Status not valid **0x00200000** IBM reserved 4 **0x00100000** IBM reserved 5 **0x00080000** IBM reserved 6 **0x00040000** IBM reserved 7 **0x00020000** IBM reserved 8 **0x00010000** IBM reserved 9 **0x00008000** IBM reserved 10 **0x00004000** IBM reserved 11 **0x00002000** IBM reserved 12	ULONG (4 bytes)

Table 53. The EGVE_PARAMETERS32 fields (continued)

Field	Description	Type (Size)
resource_flags (con't)	The flags of the selected resource. Zero, one, or more values can be set at one time. Possible values are: **0x00001000** IBM reserved 13 **0x00000800** IBM reserved 14 **0x00000400** IBM reserved 15 **0x00000200** IBM reserved 16 **0x00000100** IBM reserved 17 **0x00000080** User reserved 1 **0x00000040** User reserved 2 **0x00000020** User reserved 3 **0x00000010** User reserved 4 **0x00000008** User reserved 5 **0x00000004** User reserved 6 **0x00000002** User reserved 7 **0x00000001** User reserved 8	ULONG (4 bytes)
resource_RODM_id	The NetView RODM object ID, if applicable.	CHAR (8 bytes)
resource_status	The status of the selected resource. See the next table for a list of values that might be returned and a description of the values for the RODM `DisplayStatus` status schemes.	UNSIGNED CHAR (1 byte)
resource_status_ts	This is the resource_status timestamp.	struct EGVE_TIMESTAMP (8 bytes)
resource_TME_oid	The TME object ID, if available. Does not apply to RODM resources. Only available for instrumented resources.	PSZ (4 bytes)
resource_type	The identifier that uniquely defines the type of resource. For RODM resources, look in Display_Resource_Type_Class in RODM.	ULONG (4 bytes)

Table 53. The EGVE_PARAMETERS32 fields (continued)

Field	Description	Type (Size)
server_ip_addr	The IP address of the topology server workstation. The first 4 bytes contain 'OxFF'. The second 4 bytes is the first address in the address list of the host entry struct returned by gethostbyname().	SZ (9 bytes)
signon_username	The user name as entered on the topology console Sign On window.	PSZ (4 bytes)
user_ctrl_data	A pointer to user-determined control data. This field is used for arbitrary exit-to-exit communication.	PBYTE (4 bytes)
user_ctrl_data_len	The length of the user control data field.	USHORT (2 bytes)
user_data	A pointer to user-determined data. This field is used for arbitrary exit-to-exit communication.	PBYTE (4 bytes)
user_data_length	The length of the user data field.	USHORT (2 bytes)
user_res_data	User resource data. A pointer to resource data determined by the user. This field is used for arbitrary exit-to-exit communications.	PBYTE (4 bytes)
user_res_data_length	The length of the user_res_data field.	USHORT (2 bytes)
wait_for_cmd_response	If you want the command responses to be returned directly to the caller, and the caller is another topology server-based command exit, fill in this value with a non-zero number. The value represents the time, in minutes, for the responses to commands. Used by the IHSXTHCE exit only.	SHORT (2 bytes)
want_corr_of_cmd_rsps	Turn this field on before returning from your registered exit, if you want the topology server to send the command_string to the topology console. If this field is turned on, it is assumed you will send the responses to this command with the IHSXTJCR exit using the same correlation_id.	USHORT (2 bytes)

Table 54. Definitions of the data types (with all strings in ASCII).

Type	Description
CHAR	A character field
LONG	32-bit signed integer
PBYTE	Pointer to binary data
PSZ	Pointer to null-terminated string
SHORT	16-bit signed integer
struct	Defined in ihsiexit.h
SZ	A null-terminated string
ULONG	32-bit unsigned integer

Table 54. Definitions of the data types (with all strings in ASCII). (continued)

Type	Description
USHORT	16-bit unsigned integer

Table 55 maps the resource status values returned in the resource_status field in EGVE_PARAMETERS32 to the descriptions for the RODM DisplayStatus status schemes.

Table 55. Descriptions of the resource_status values in EGVE_PARAMETERS32

resource_status Value in EGVE_PARAMETERS32	RODM DisplayStatus Description
0	Satisfactory
2	Medium satisfactory
5	Low satisfactory
6	Intermediate
9	Degraded
12	Low unsatisfactory
14	Severely degraded
17	Medium unsatisfactory
18	Unsatisfactory
20	Unknown
21	Deleted
24	Scheduled
25	_IBM positive 4
26	_IBM positive 5
27	_IBM positive 6
28	_IBM positive 7
29	_IBM positive 8
30	_IBM negative 3
31	_IBM negative 4
32	_IBM negative 5
33	_IBM negative 6
34	_IBM negative 7
35	_IBM negative 8
36	_User positive 1
38	_User positive 2
39	_User positive 3
40	_User positive 4
41	_User positive 5
42	_User positive 6
43	_User positive 7
44	_User positive 8
45	_User negative 1

Table 55. Descriptions of the resource_status values in EGVE_PARAMETERS32 (continued)

resource_status Value in EGVE_PARAMETERS32	RODM DisplayStatus Description
47	_User negative 2
48	_User negative 3
49	_User negative 4
50	_User negative 5
51	_User negative 6
52	_User negative 7
53	_User negative 8

Table 56 maps the status of RODM resources into the resource_status field in EGVE_PARAMETERS32. The RODM resource value is stored in the DisplayStatus field.

Table 56. The RODM DisplayStatus field and the resource_status value in EGVE_PARAMETERS32

DisplayStatus Value in RODM	resource_status Value in EGVE_PARAMETERS32
129	0
130	18
131	6
132	20
133	9
134	14
135	21
136	36
137	38
138	39
139	40
140	41
141	42
142	43
143	44
144	2
145	5
146	24
147	25
148	26
149	27
150	28
151	29
152	45

Table 56. The RODM DisplayStatus field and the resource_status value in EGVE_PARAMETERS32 (continued)

DisplayStatus Value in RODM	resource_status Value in EGVE_PARAMETERS32
153	47
154	48
155	49
156	50
157	51
158	52
159	53
160	17
161	12
162	30
163	31
164	32
165	33
166	34
167	35

Command Exit Functions for C

This section describes the command exit interface functions that are available from the C programming environment. Function prototypes are provided in the ihsiexit.h C include file.

IhsiInit - Register a Command Exit

Function Ihsilnit defines a command exit and its associated command exit procedure to the command exit facility. When the command exit is driven, the command exit procedure will be driven with a parameter pointing to the EGVE_PARAMETERS32 structure.

Example:

```
unsigned int IhsiInit(char * exit_name,
EGVE_EXIT_PROC_ADDR32 exit_procedure_addr, int thread_type);
```

Where:

exit_name
: Null-terminated string (maximum 8 characters) that specifies the name of the command exit to be registered. The *exit_name* must match the name defined in the CPE Exit name field.

exit_procedure_addr
: Address of the command exit procedure to be driven when *exit_name* is invoked.

thread_type
: Specifies the type of threading to use. Possible values are:

  - EGVE_SINGLE_THREADING: multiple executions of this exit are serialized under one thread.

- EGVE_MULTI_THREADING: multiple executions of this exit are invoked under multiple threads.

Possible return values are listed in “Return Codes from Command Exit Interface Functions” on page 120.

IhsiInitialize - Register a Command Exit

This function defines a command exit and its associated command exit procedure to the command exit facility. When the command exit is driven, the command exit procedure is driven with a parameter pointing to the EGVE_PARAMETERS32 structure.

Using this initializing routine defaults to a thread_type of EGVE_MULTI_THREADING. See “IhsiInit - Register a Command Exit” on page 117 for further information.

Example:

```
unsigned int IhsiInitialize(char * exit_name,
EGVE_EXIT_PROC_ADDR32 exit_procedure_addr);
```

Where:

exit_name	Null-terminated string (maximum 8 characters) that specifies the name of the command exit to be registered. The *exit_name* must match the name defined in the CPE Exit name field.
exit_procedure_addr	Address of the command exit procedure to be driven when *exit_name* is invoked.

Possible return values are listed in “Return Codes from Command Exit Interface Functions” on page 120.

IhsiSend - Remotely Invoke a Command Exit

IhsiSend enables a C command exit to perform exit-to-exit invocations. This function makes a remote procedure call to another command exit procedure (possibly on another workstation) using the command exit facility. The parameters in the EGVE_PARAMETERS32 structure, and the data to which those parameters point, are collected and transferred to the destination workstation.

The command exit on the destination workstation is driven. When the exit function is complete, it performs a return statement passing a return value. The IhsiSend function call then completes with a return code from the invoked command exit.

Example:

```
unsigned int IhsiSend(char * source_exit_name, char * dest_exit_name,
char * dest_ip_addr, EGVE_PARAMETERS32_PTR parms_ptr);
```

Where:

source_exit_name	Null-terminated string (maximum 8 characters) that specifies the name of the command exit performing this IhsiSend.
dest_exit_name	Null-terminated string (maximum 8 characters) that specifies the name of the command exit to be invoked.
dest_ip_addr	Null-terminated string (maximum 8 characters) that

	specifies the IP address of the topology server workstation where this command exit is to be invoked. This is the IP address from the perspective of the topology server.
parms_ptr	Pointer to the EGVE_PARAMETERS32 structure to be sent to *dest_exit_name*.

Usage:

The IhsiInitialize or IhsiInit function must be performed to register *source_exit_name* before calling IhsiSend.

IhsiSend must be invoked by a function that was invoked by a command exit. You cannot write a stand-alone program that issues the IhsiSend function call. can only be invoked from a registered command exit procedure because certain fields in the EGVE_PARAMETERS32 structure passed to IhsiSend are available only when a function is invoked as a command exit. These fields must be passed along to any other invoked command exit.

A command exit can issue more than one IhsiSend function call.

If the IhsiSend function is not able to invoke the command exit specified by the *dest_exit_name* parameter, possible return values are listed in “Return Codes from Command Exit Interface Functions” on page 120.

If IhsiSend is able to invoke the command exit, the return value is one of those listed in “Other Return Codes” on page 121. In this case, the return value is an exit-to-exit return code.

IhsiTerminate - Unregister a Command Exit

This function unregisters a command exit and its associated command exit procedure from the command exit facility. The command exit facility has no further knowledge of this command exit until another registration (IhsiInitialize or IhsiInit) is performed.

Example:

```
unsigned int IhsiTerminate(char * exit_name);
```

Where:

exit_name	Null-terminated string (maximum 8 characters) that specifies the name of the command exit to be unregistered.

Possible return values are listed in “Return Codes from Command Exit Interface Functions” on page 120.

IhsiWait - Wait Before Terminating

This function causes the command exit to wait until the topology communications server terminates or specifies the length of time an exit should wait before terminating. This function must be called after IhsiInitialize or IhsiInit and before IhsiTerminate. It is recommended that command exits monitor this signal and terminate when the topology server terminates. This frees system resources, preventing the user from having to manually terminate multiple command exits and enabling the maintenance of command exit executable files.

Example:

```
unsigned int IhsiWait(char * exit_name, int seconds);
```

Where:

exit_name — Null-terminated string (maximum 8 characters) that specifies the name of the command exit to wait.

seconds — The length of time to wait.

- -1 specifies an indefinite wait (until the topology communication topology server stops).
- 0 specifies no wait.
- Any number greater than 0 specifies the number of seconds to wait.

Possible return values are listed in "Return Codes from Command Exit Interface Functions".

Return Codes

Return Codes from Command Exit Interface Functions

Table 57 describes the return values from the IhsiInitialize, IhsiInit, IhsiTerminate, IhsiSend or IhsiWait functions. These return values are interface codes. A value other than EGVE_OK indicates an unsuccessful completion of the command exit interface function.

Table 57. Return codes from command exits IhsiInitialize, IhsiInit, IhsiTerminate, IhsiSend, and IhsiWait

Return Value	Code	Description
EGVE_OK	0x0000	The interface function completed successfully.
EGVE_ALREADY_STARTED	0x0001	The exit name specified is already registered.
EGVE_PARAMETER_CHECK	0x0002	A required parameter to one of the command interface functions was missing or not valid; or an unused field in the EGVE_PARAMETERS32 structure was not set to NULL.
EGVE_OUT_OF_MEMORY	0x0003	The command exit facility could not obtain memory to invoke the exit.
EGVE_NOT_STARTED	0x0004	A command exit interface function was called before an IhsiInitialize or IhsiInit was performed.
EGVE_HANDLER_NOT_STARTED	0x0005	Topology communication server has not been started correctly or is not running.
EGVE_DATA_OVERFLOW	0x0006	The total length of the data pointed to by the EGVE_PARAMETERS32 pointers exceeds 32768 bytes. Reduce the length of the user_data or user_ctrl_data or user_res_data fields.

Table 57. Return codes from command exits IhsiInitialize, IhsiInit, IhsiTerminate, IhsiSend, and IhsiWait (continued)

Return Value	Code	Description
EGVE_INTERNAL_FAILURE	0x0007	An unrecoverable error occurred in a command exit interface function.
EGVE_SEM_TIMEOUT	0x0008	When invoking IhsiWait, the time specified has expired.
EGVE_SEM_TERMINATE	0x0009	When invoking IhsiWait, the topology communication server is shutting down.

Other Return Codes

Table 58 describes the values returned by an exit. These values will be returned to one of these:

- The exit that called this exit
- The topology server, if this was the first exit invoked

The return values from EGVE_EXIT_USER through EGVE_EXIT_USER_MAX are available as user-defined return codes.

Table 58. Return codes from command exits

Return Value	Code	Description
EGVE_EXIT_OK	0x0000	The command exit completed successfully.
EGVE_EXIT_INVALID_NV_DOMAIN	0x801A	The NetView domain ID contains invalid syntax. The domain ID must meet the following criteria. • It must be from 1 to 5 characters in length. • The first character must be alphabetic (either upper or lower case) or one of the following characters: @#$. • The remaining characters (second through fifth) must be alphabetic (either upper or lower case), numeric, or one of the following characters: @#$.
EGVE_EXIT_NOT_LOADED	0x8001	The exit name, specified by the dest_exit_name parameter of the IhsiSend function, could not be loaded. **Possible causes**: • The dest_exit_name executable file could not be found. • An IhsiInitialize or IhsiInit was not performed for dest_exit_name.
EGVE_EXIT_FAIL	0x8002	The command exit invoked was not able to complete its function due to an error.
EGVE_EXIT_NO_OPER	0x8004	The operator ID specified in the command exit parameter block is not signed on to the NetView management console or the NetView host.

Table 58. Return codes from command exits (continued)

Return Value	Code	Description
EGVE_EXIT_INVALID_CMD	0x8005	The command string specified in the command exit parameter block is not valid; for example, the command is too long.
EGVE_EXIT_HOST_DOWN	0x8006	The command exit facility is unable to communicate with the NetView host. The NETCONV session between the topology server and the NetView host is not active.
EGVE_EXIT_NO_DOMAINS	0x8007	When a SNA Topology Manager resource is selected, a NetView domain name could not be determined. When the HOSTCMD is run, the TCP/IP hostname could not be converted to a NetView domain name.
EGVE_EXIT_OUT_OF_MEMORY	0x8009	The command exit did not obtain memory to complete its function.
EGVE_EXIT_OUT_OF_RESOURCES	0x800a	The command exit facility could not obtain a resource to invoke a command exit; for example, a thread could not be started.
EGVE_EXIT_INVALID_DEFINITION	0x800b	The data passed to the command exit was either not valid or was inconsistent with the requirements of the exit. Possible errors are: • The command string was not valid or not defined. • The exit required resource information, but the exit was defined as resource-independent. • The particular resource type passed to an exit is not supported by that exit. • The command defined to CPE references a DDF file, but the DDF file could not be located in the DPATH environment variable.
EGVE_EXIT_TME_TASK_NOT_FOUND	0x800c	The TME task was not found. Only returned by the IHSMTTME command exit.
EGVE_EXIT_TME_CATCH_IN_MAIN	0x800d	An unexpected exception was thrown while invoking the TME task. Only returned by ISHMTTME command exit.
EGVE_EXIT_TME_INVALID_MANAGEDNODE_NAME	0x800e	The hb_hostname field in the EGVE_PARAMETERS32 block is null and a TME managed node name is expected. Only returned by the IHSMTTME command exit.
EGVE_EXIT_TME_INVALID_COMMAND_STRING	0x800f	The syntax of the command string input is not correct. It did not follow the format of tasknameltaskLibraryName. Only returned by the IHSMTTME command exit.
EGVE_EXIT_TME_ERR_GET_ALL_RESOURCES	0x8010	The TME exception was thrown during the Get All Resources (such as administrator, managed nodes, and so on) processing. Only returned by the IHSMTTME command exit.

Table 58. Return codes from command exits (continued)

Return Value	Code	Description
EGVE_EXIT_TME_BAD_RETURN_FROM_TAS	0x8011	The IHSMTTME command exit received a non-zero return code from the IhsiSend to the IHSMTTAS command exit. The IHSMTTAS is an IBM internal use only command exit that actually executes the task. Only returned by the IHSMTTME command exit.
EGVE_EXIT_TME_ERROR_DURING_TASK_EXEC	0x8012	An error occurred during the execution of a TME task. This return code is returned from the IHSMTTAS IBM internal use only command exit to IHSMTTME command exit. However, the IHSMTTME command exit will not return this return code to the calling exit but will return EGVE_EXIT_TME_BAD_RETURN_FROM_ TAS.
EGVE_EXIT_OPER_NOT_AUTH	0x8019	The operator ID specified in the command exit parameter block is not authorized to issue NetView commands.
EGVE_EXIT_USER	0x9000	This is the start of the range of command exit return codes available to the user for user-written exit-to-exit communication. The end of this range is specified by EGVE_EXIT_USER_MAX.
EGVE_EXIT_USER_MAX	0x9fff	This is the end of the range of command exit return codes available to the user for user-written exit-to-exit communication. The start of this range is specified by EGVE_EXIT_USER.

Invoking Command Exits with a C Interface

IHSDNATV Command Exit

The IHSDNATV command exit sends a command to GMFHS, which forwards the command to the service point for the specified resource. This is done by filling in the appropriate parameters in the EGVE_PARAMETERS32 structure and then calling the IhsiSend function.

All GMFHS commands are resource-dependent and require that the resource_RODM_id field be set in the EGVE_PARAMETERS32 structure. Table 59 illustrates the C interface to the IHSDNATV command exit.

Table 59. C interface to command exit IHSDNATV

Field	Description	Type (Size)
client_handle	The client_handle field must be the same as the client_handle field in EGVE_PARAMETERS parameter block of the command exit invoking IHSDNATV.	ULONG (4 bytes)

Table 59. C interface to command exit IHSDNATV (continued)

Field	Description	Type (Size)
client_ip_addr	Set this field to the same value as that passed to you in the EGVE_PARAMETERS32 structure when your exit was invoked.	SZ (9 bytes)
command_string	The command string to send to GMFHS.	PSZ (4 bytes)
correlation_id	Set this field to the same value that is passed to you in the EGVE_PARAMETER32 structure when your exit was invoked.	struct EGVE_CORR_ID (4 bytes)
display_name	Set this field to the same value that is passed to you in the EGVE_PARAMETER32 structure when your exit was invoked.	PSZ (4 bytes)
op_id	The operator ID that has signed on to the NetView host	SZ (9 bytes)
resource_RODM_id	This field is the 8 bytes of a RODM object ID.	CHAR (8 bytes)
resource_type	Set this field to the same value that is passed to you in the EGVE_PARAMETER32 structure when your exit was invoked.	ULONG (4 bytes)
server_ip_addr	Set this field to the same value as that passed to you in the EGVE_PARAMETERS32 structure when your exit was invoked.	SZ (9 bytes)

Note: For data type definitions, see Table 54 on page 114.

IHSXTHCE Command Exit

A user-written command exit can invoke the IHSXTHCE command exit to execute a command to the NetView host. This is done by filling in the appropriate parameters in the EGVE_PARAMETERS32 structure and then calling the IhsiSend function.

Resource-independent commands should set the res_IBM_id field in the EGVE_PARAMETERS32 structure to zero. See “IHSXTHCE Command Exit” on page 101 and “Substitution Variables” on page 102 for information about command string substitutions.

The dest_ip_addr parameter in the IhsiSend call must be set to the value in the server_ip_addr field of the EGVE_PARAMETERS32 structure that was passed to the exit invoking IHSXTHCE.

The IHSXTHCE command exit is driven on the topology server workstation. After IHSXTHCE sends the command to the host, the IhsiSend will complete. The return value from IhsiSend indicates the success or failure of getting the command to the NetView host for execution; the return value does not indicate the success or failure of the command execution to the NetView host.

Note: To suppress command responses, include %noresponse% in the command string.

Table 60 illustrates the C interface to the IHSXTHCE command exit.

Table 60. C interface to command exit IHSXHCE

Field	Description	Type (Size)
client_ip_addr	Set this field to the same value as that passed to you in the EGVE_PARAMETERS32 structure when your exit was invoked.	SZ (9 bytes)
command_indicator	Set this field to the same value as that passed to you in the EGVE_PARAMETERS32 structure when your exit was invoked.	USHORT (2 bytes)
command_string	The command string to send to the host. Example: (V NET,ACT,ID=RESOURCE). A command_string of length 0 is treated as a request to unlock a locked screen (indicator ***). It is not treated as an Enter key. The command_string cannot be longer than 255 characters, including any expansion of substitution variables.	PSZ (4 bytes)
correlation_id	Set this field to the same value as that passed to you in the EGVE_PARAMETERS32 structure when your exit was invoked.	struct EGVE_CORR_ID (4 bytes)
data_source	Set this field to the same value as that passed to you in the EGVE_PARAMETERS32 structure when your exit was invoked.	USHORT (2 bytes)
display_name	Set this field to the same value that is passed to you in the EGVE_PARAMETER32 structure when your exit was invoked.	PSZ (4 bytes)
nv390_domain	Set this field to the NetView domain ID where the command is to be run. The command will be routed from the NetView host that is connected to the topology server to this NetView domain for execution. If a value of all blanks or nulls is used, then the default is to execute the command on the NetView host to which the topology server is connected. This field is not used when the IHSXTHCE command exit is called by the sample IHSSTHSTC command exit (used by tserver hostcmd).	SZ (6 bytes)
op_id	The operator ID that has signed on to the NetView host.	SZ (9 bytes)

Table 60. C interface to command exit IHSXHCE (continued)

Field	Description	Type (Size)
res_IBM_id	Set this field to the same value as that passed to you in the EGVE_PARAMETERS32 structure when your exit was invoked. If the command is resource-dependent, this field must be non-zero. The topology server then uses the res_IBM_id to gather resource specific information for command string substitutions. If res_IBM_id is zero, this command is not executed against any particular resource because it is resource-independent.	ULONG (4 bytes)
resource_RODM_id	Set this field to the same value that is passed to you in the EGVE_PARAMETER32 structure when your exit was invoked.	CHAR (8 bytes)
resource_status	Set this field to the same value that is passed to you in the EGVE_PARAMETER32 structure when your exit was invoked.	UNSIGNED CHAR (1 byte)
server_ip_addr	Set this field to the same value as that passed to you in the EGVE_PARAMETERS32 structure when your exit was invoked.	SZ (9 bytes)
user_data	If you want the command responses returned directly to your command exit, this field must point to the storage where the responses will be written.	PBYTE (4 bytes)
user_data_length	The length of the storage pointed to by user_data.	USHORT (2 bytes)
wait_for_cmd_response	Time, in minutes, to wait for a command response. A value of zero means that you do not wait. Make sure you do not have a %NORESPONSE% in the command string. If you want the command responses you must allocate enough space to hold them in user_data. The responses are mapped by the EGVE_CMD_RSP structure.	SHORT (2 bytes)

Note: For data type definitions, see Table 54 on page 114.

IHSXTJAV Command Exit

The IHSXTJAV command exit invokes a Java class on the topology console. Call this command exit from your command exit through IhsiSend to launch your Java class on the topology console. See “Chapter 6. Topology Console Java Applications and Plug-ins” on page 51 for information about installing these Java classes.

Table 61 illustrates the C interface to the IHSXTJAV command exit.

Table 61. C interface to command exit IHSXTJAV

Field	Description	Type (Size)
client_handle	Set this field to the same value that is passed to you in the EGVE_PARAMETER32 structure when your command exit was invoked.	ULONG (4 bytes)
client_ip_addr	Set this field to the same value as that passed to you in the EGVE_PARAMETERS32 structure when your command exit was invoked.	SZ (9 bytes)
command_string	Set the first blank delimited token to be the Java class name to be launched on the topology console.	PSZ (4 bytes)
server_ip_addr	Set this field to the same value as that passed to you in the EGVE_PARAMETERS32 structure when your command exit was invoked.	SZ (9 bytes)

Note: For data type definitions, see Table 54 on page 114.

IHSXTJCR Command Exit

The IHSXTJCR command exit sends a command response from the topology server back to the topology console, where it will be displayed in the topology console log area. Call this command exit from your command exit through IhsiSend to send command responses to the topology console.

Table 62 illustrates the C interface to the IHSXTJCR command exit.

Table 62. C interface to command exit IHSXTJCR

Field	Description	Type (Size)
client_handle	Set this field to the same value that is passed to you in the EGVE_PARAMETER32 structure when your exit was invoked.	ULONG (4 bytes)
client_ip_addr	Set this field to the same value as that passed to you in the EGVE_PARAMETERS32 structure when your exit was invoked.	SZ (9 bytes)

Table 62. C interface to command exit IHSXTJCR (continued)

Field	Description	Type (Size)
correlation_id	A unique number used to correlate requests and responses. This field in the EGVE_PARAMETERS32 structure should not be modified by a user command exit. Altering this field could result in an error when this exit returns to its invoker. This number is used by the topology server to correlate the command to the command response. The command string is sent by the topology server to the topology console if you set want_corr_of_cmd_rsps before returning from your registered exit procedure. Then, using the IhsiSend API, you can send responses to your command by calling IHSXTJCR using the same correlation number.	struct EGVE_CORR_ID (4 bytes)
server_ip_addr	Set this field to the same value as that passed to you in the EGVE_PARAMETERS32 structure when your command exit was invoked.	SZ (9 bytes)
user_ctrl_data	Pointer to structure EGVE_CMD_RSP_CTRL.	PBYTE (4 bytes)
user_ctrl_data_length	Size of structure EGVE_CMD_RSP_CTRL.	USHORT (2 bytes)
user_data	Pointer to an array of 1 to n serialized EGVE_CMD_RSP structures. Can be null if user_data length is zero. These structures contain the command responses.	PBYTE (4 bytes)
user_data_length	Size of the serialized structures pointed to by user_data.	USHORT (2 bytes)

Note: For data type definitions, see Table 54 on page 114.

Determining Additional Resource Information

A command exit can determine the resource manager that manages a resource, whether a resource is real or aggregate.

Resource Manager Determination

A command exit can use the command indicator values defined with the command profile editor to see which resource manager manages the resource. See “Chapter 9. Using the NMC Command Profile Editor” on page 87 for information about command indicators.

Real or Aggregate Determination

To determine if this is a real or aggregate resource, add the statement `#include "ihsuduix.h"` at the top of your source file. This file is located in one of the following:

- For Intel: %BINDIR%\TDS\server\sample

- For UNIX: `$BINDIR/TDS/server/sample`

Use the resource_type field in the EGVE_PARAMETERS32 block. If the expression `(resource_type & DUIXC_MASK_AGG_FLAG == DUIXC_MASK_AGG_FLAG)` evaluates to true, the resource type is an aggregate; otherwise, the resource is real.

Chapter 11. Converting NGMF Command Sets

The NetView resource-specific commands that are part of the MultiSystem Manager (MSM) component which shipped with the NetView Graphic Monitor Facility (NGMF) have been converted for use with the NetView management console, and the required files have been shipped with NetView management console. If you have created NGMF command sets, convert them so they can be used with the NetView management console.

To convert an NGMF command set for use with the NetView management console, perform the following steps:

1. Convert the NGMF response file (.RSP).

   See “Converting the NGMF Response File” on page 132 for details.
2. Convert the NGMF command tree facility definition file (.CDF).

   See “Converting the NGMF Command Tree Facility Definition File” on page 133 for details.
3. Combine the output files from tasks 1 and 2 into a single NetView management console response file (.RSP).

   See “Combining the Output Files” on page 136 for details.
4. If your command sets reference dialog definition files (.DDF), copy your existing DDF files from the NGMF directory to the topology server directory. These are copied without modification to one of the following directories:
   - For Intel: %BINDIR%\TDS\server\config\ddf\xxx
   - For UNIX: $BINDIR/TDS/server/config/ddf/xxx

   Note: The variable '*xxx*' indicates the country code, such as ″en_US″ or ″ja_JP.″

If you customized the NetView resource-specific commands, some conversion tasks are necessary. If you customized the NGMF response file (.RSP), but not the NGMF command tree facility definition file, you must perform tasks 1, 2, and 3. Obtain your customized NGMF response file as input to task 1. You must get the corresponding NGMF command tree facility definition file (.CDF) from the installed NetView resource-specific command sets and use it as input to task 2. The command tree facility definition file names are contained in Table 63.

Table 63. Command tree facility definition files

Feature	Command Tree Facility Definition File	Converted NGMF Response File
ATM	FLCA0001.CDF	FLC001AR.RSP
NetWare	FLCE0001.CDF	FLC001ER.RSP
NetFinity	FLCH0001.CDF	FLC001HR.RSP
IP	FLCI0001.CDF	FLC001IR.RSP
LNM	FLCL0001.CDF	FLC001LR.RSP
Open	FLCO0001.CDF	FLC001OR.RSP
TMR	FLCT0001.CDF	FLC001TR.RSP

If you customized the NGMF command tree facility definition file (.CDF), but not the NGMF response file (.RSP), you must perform tasks 2 and 3. Use your customized

NGMF command tree facility definition file as input to task 2. The already-converted NGMF response files are shipped with the NetView management console and correspond to the output of task 1. These files are installed into the same directory as the NetView management console command profile editor and cpebatch utility. The converted file names are contained in Table 63 on page 131.

If you have customized both the response file (.RSP) and the component description file (.CDF), you must perform tasks 1, 2, and 3. Use your customized response file as the input to task 1. Use your customized component description file as input to task 2.

Converting the NGMF Response File

The NGMF response file must be converted manually through the use of an ASCII text editor. Copy the NGMF response file, `FLCxR001.RSP`, to a file named `FLC001xR.RSP` (for example, `FLCAR001.RSP` to `FLC001AR.RSP`), then edit the copied file as follows:

1. Delete all lines in the command block containing:

   ```
   NAME = Resource specific commands
   ```

   from:

   ```
   COMMAND = (
   ```

   to the end of that command block with the matching) tag.
2. Delete all lines in the command blocks containing the following:

   ```
   NAME = Non-SNA command line
   NAME = Remote console
   ```

   from:

   ```
   COMMAND = (
   ```

   to the end of the command blocks with the matching) tags.
3. Change all commands with the following line:

   ```
   EXIT_NAME = DUICGENE
   ```

   to:

   ```
   EXIT_NAME = IHSDGENE
   ```
4. Add the following lines to the end of the remaining command page block:

   Note: These two lines must be inside the command page block.

   ```
   CLIENT_PLATFORM_LIST = GENERIC
   TARGET_PLATFORM_LIST = GENERIC
   ```
5. Remove the following line from all commands:

   ```
   LU_NAME = *
   ```
6. For the Alert history command, change the following line:

   ```
   NAME = Alert history
   ```

   to:

   ```
   NAME = Event Viewer
   ```

   Also, for the Alert history command, change the following lines:

   ```
   COMMAND_STRING =
   EXIT_NAME = DUICALRT
   ```

to:

```
COMMAND_STRING = com.tivoli.ihs.client.eviewer.IhsEventViewer
EXIT_NAME = IHSXTJAM
```

Converting the NGMF Command Tree Facility Definition File

Before converting the NGMF command tree facility definition file (.CDF), gather the information described in the following text.

Determine the resource manager name which is listed in the NGMF response file as Manager Name. The prefix used for the help files is FLC (for the NetView resource-specific commands). The help panels are in individual files, not in a single, large file.

Also, determine the command indicators to be used in this conversion. These command indicators are defined in the NGMF response file. The command indicators that are used for each command subset are required.

To determine the command indicators for each subset, use an ASCII editor to browse the original NGMF response file, prior to performing any conversions, and perform the following steps:

1. Find the beginning of the Resource specific commands block.
2. For each page in the Resource specific commands block, find the COMMAND_STRING line. In that line, find the SUBSET keyword. The name assigned to that subset is the name that is matched with a name in the side tabs in the conversion utility dialog where the command indicators are entered. An example of a subset name is ATM_Switch.
3. In the same page, find the INDICATOR_LIST line. In the INDICATOR_LIST block, you will find keywords, such as VALUE.0, VALUE.1, and so on. The values assigned to these keywords are the command indicators corresponding to the subset identified in step 2. If the values are numeric (for example, 34409), record the list of values with the subset name from step 2.
4. If the values found in step 3 are defined constants, such as INDICATOR_LOW, INDICATOR_LOW+47, and so on, convert them to numeric values. Find the number corresponding to INDICATOR_LOW in the MANAGER definition block near the top of the response file. For example, if the INDICATOR_LOW line is INDICATOR_LOW = 34408, then the numeric value of INDICATOR_LOW is 34408. The numeric value of INDICATOR_LOW+47 is 34455, and so on. Record the list of numeric values with the subset name from step 2.
5. Repeat steps 2 and 3 and (if necessary) 4, until you have recorded the subset name and command indicators for each page in the resource-specific commands block.

   If there is a subset that is listed on the conversion utility dialog, but is not listed in any page of the resource-specific commands in the NGMF response file, there are no command indicators for that subset.

When you have the previous information, follow these steps to convert an NGMF command tree facility definition file (.CDF(to a command profile editor response file (.RSP):

1. Change to the topology console bin directory:
   - For Intel: %BINDIR%\..\generic_unix\TDS\client\bin
   - For UNIX: $BINDIR/../generic_unix/TDS/client/bin

   where BINDIR is the install directory of the topology console.

2. Enter the following command to start the conversion utility:

   ```
   tappxx .. com.tivoli.ihs.client.cmdtree.IhsCT2RSPUtil CDF RSP
   ```

   Where xx is the appropriate platform from which the topology console is running. CDF must be fully qualified to the path where the CDF files have been installed (usually located by `DPATH`). See “Appendix B. Topology Console Commands” on page 161 for more information about the **tapp***xx* command.

   Note: To download all of the support files necessary for **tapp***xx* to be available (such as the online help files) from the NetView management console Server, sign on using the main NMC Console at least one time.

   The window shown in Figure 24 is displayed.

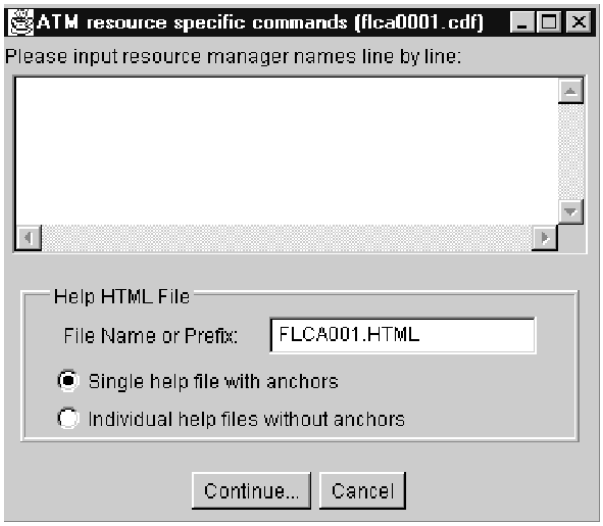


Figure 24. ATM Resource Specific Commands - Resource Manager Names Dialog

3. In the upper text field, enter the names of all resource managers that will be using the commands and command sets. Also, specify the following:
   - In the File Name or Prefix text field, the file name or prefix of the HTML help file for the command tree facility commands.
   - Whether the help files should be contained in one flat file or that a different help file should be created for each command by selecting one of the following in the Help HTML File section:
     – **Single help file with anchors**, and specify the CommandFileName.HTML file.
     – **Individual help files without anchors**, and specify a prefix for these files. It is recommended that you use a 3-character prefix.
4. Click **Continue**. The screen shown in Figure 25 on page 135 is displayed.

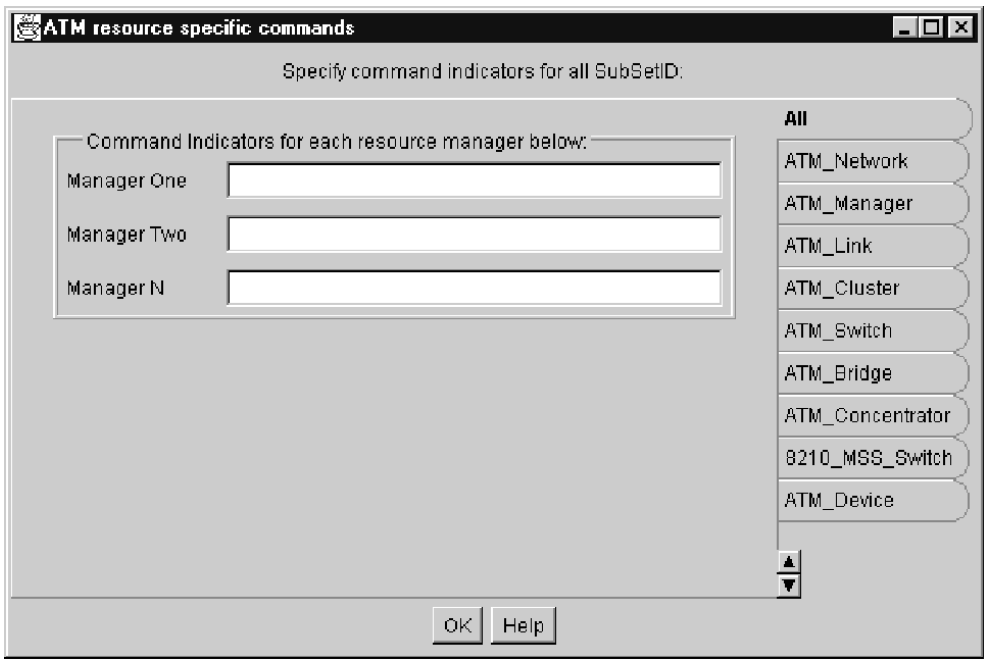


Figure 25. ATM Resource Specific Commands - Command Indicators Dialog

5. From the resource specific commands - Command Indicators dialog, specify the command indicators for each resource manager in the **Manager One**, **Manager Two** and **Manager N** fields.
6. Click **OK** when all indicators have been entered. The screen shown in Figure 26 is displayed.

Figure 26. ATM Resource Specific Commands - Verification Dialog

7. Click **Yes** to verify the command indicators or to change help files. Click **No** if you want to create the command profile editor response file without verifying the command indicators.
8. If you choose to verify the command indicators, dialogs are presented in Figure 27 on page 136 for each command belonging to a command set and that command set name. Verify the values and make appropriate changes. When you have verified the command indicators, click **OK** to create the response file.

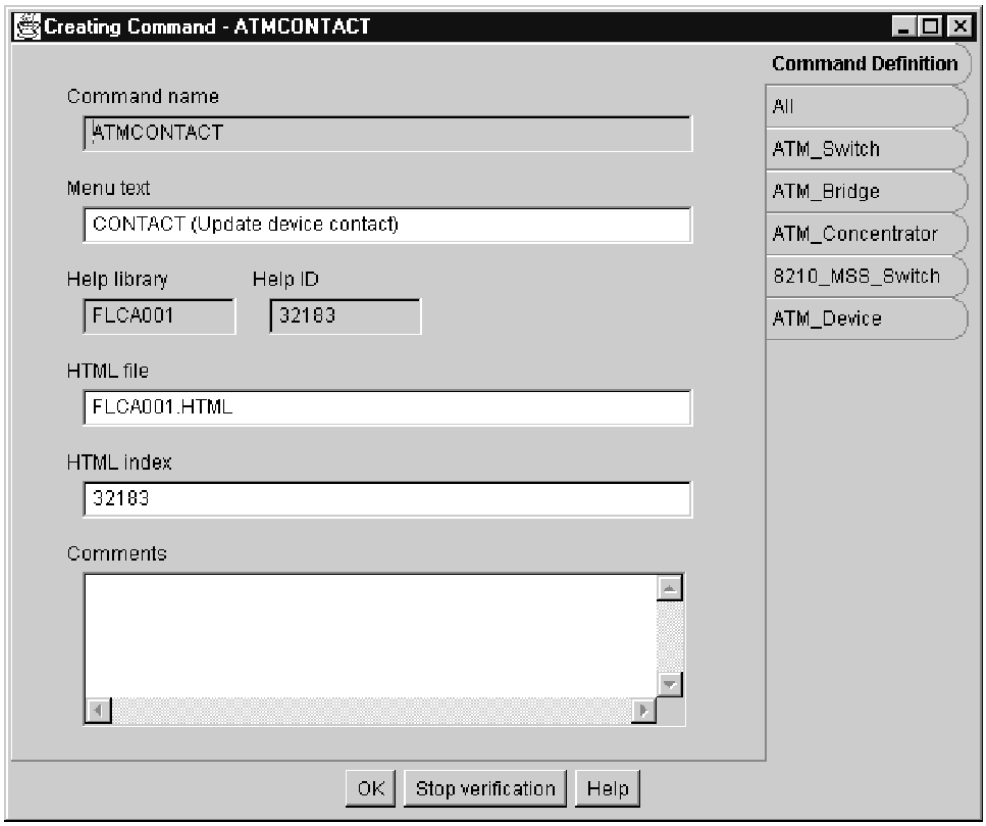


Figure 27. Creating Command Dialog

If, at any time during this process, you are satisfied with the verification process or choose not to verify the rest of the commands or command sets, you can select **Stop verification**, and the response file will be created.

Combining the Output Files

To combine the two output files created in Figure 28 on page 137 into a single CPE response file (.RSP), copy the two response files into a third response file.

The following is an example of the **copy** command, which should be issued from the command prompt:

```
copy FLC001AR.RSP+FLCA0001.RSP FLCA001N.RSP
```

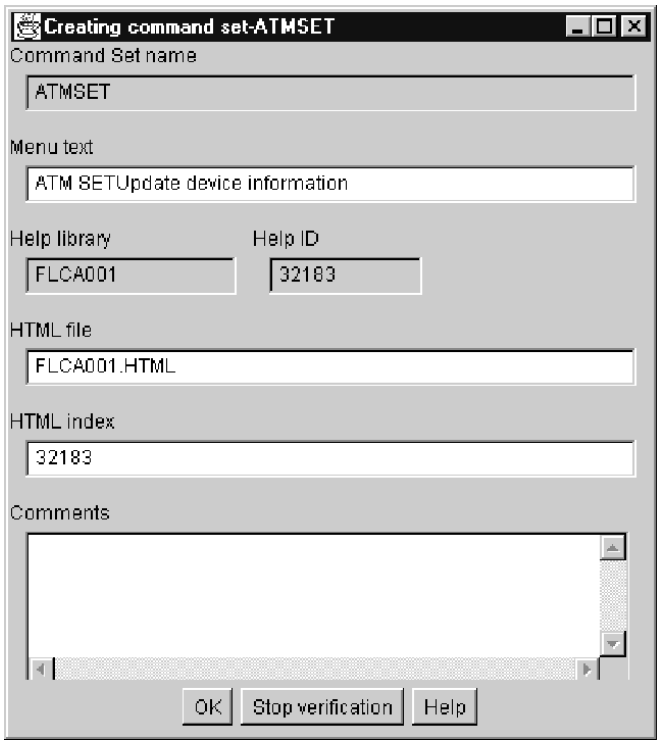


Figure 28. Creating Command Set Window

Where:

- `FLC001AR.RSP` is the output file from Task 1.
- `FLCA0001.RSP` is the output file from Task 2.
- `FLCA001N.RSP` is the single response file to be used by the NetView management console.

The CPE response file (.RSP) is the file used by the **cpebatch** command to load the command profile editor database. For more information on the **cpebatch** command, see “cpebatch” on page 144.

Part 4. Appendixes

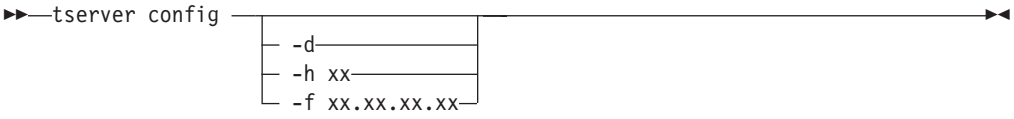
Appendix A. Topology Server Commands

The topology server commands provided in this appendix are intended to be used as reference material. Syntax diagrams are provided for each command.

config

Format

config



Purpose

The **config** command enables you to specify that the topology server processes are to be started as daemons. This command also enables you to specify the heartbeat interval for the topology server.

Parameters

-d Specifies that the topology server processes be started as daemons. Updates are made to the appropriate system files so processes start automatically when the system is started.

This operand can be used only on the UNIX platform.

-h xx
Enables you to specify the heartbeat interval for the topology server. The xx specifies the interval, in minutes, for generating heartbeats, such that the topology server resource in the topology display subsystem view remains in satisfactory status. The default is 5 minutes.

-f xx.xx.xx.xx
Enables you to specify an additional address by which your machine is known. This can be used if you are using network address translation (NAT) and the topology server is being accessed by this method.

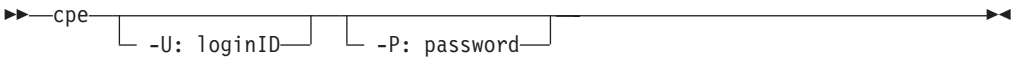
The variable, *xx.xx.xx.xx*, is the TCP/IP address in dot notation.

Note: The server must be rebooted after issuing this command in order for it to take effect.

cpe

Format

cpe



Parameters

-U *loginID*

Enter a valid login user ID. The user ID must match your NetView user ID.

The login user ID must have administrative authority.

-P *password*

The password for the login user ID specified by the -U parameter. There is no default value. This password must match your NetView password. If a value is not specified, the command profile editor utility tries to sign on to the topology server with a null password. If this fails, the command profile editor Sign On window is displayed. This is an optional parameter.

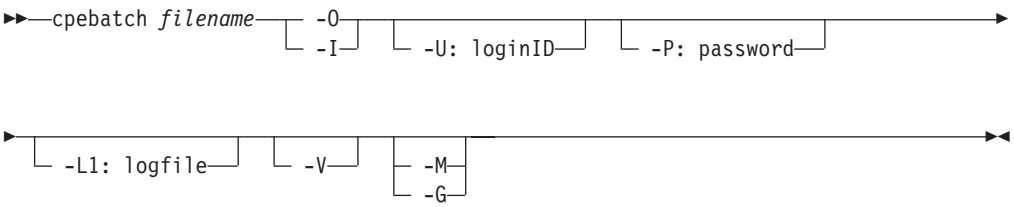
Purpose

Starts the command profile editor GUI facility. This is available on Intel platforms only. For additional information about the command profile editor, see “Chapter 9. Using the NMC Command Profile Editor” on page 87.

cpebatch

Format

cpebatch



Purpose

Starts the command profile editor batch utility. For additional information about the command profile editor, see “Chapter 9. Using the NMC Command Profile Editor” on page 87.

Parameters

cpebatch
The name of a command that invokes the executable command profile editor utility program.

filename
This is a required parameter. If the -O option is specified, this is the name of the output file that is created by the utility. If the -I option is specified, this is the name of the input file that is read by the utility.

-O
Generates a response file from the current commands database. The *filename* specifies the name of the file generated by the utility. If this is not a fully-qualified name, the file is generated in the current directory. If no name is specified, the default name of `IHSECPED.RSP` is used. The -O or -I parameter must be specified.

-I Specifies a response file to be used to update the current commands database. Unless the filename is fully-qualified, it is assumed to reside in the current directory. The -O or -I parameter must be specified.

-U *loginID*

Enter a valid login user ID. The user ID must match your NetView user ID.

The login user ID must have administrative authority.

-P *password*
The password for the login user ID specified by the -U parameter. There is no default value. This password must match your NetView password. If a value is not specified, the command profile editor utility tries to sign on to the topology server with a null password. If this fails, the command profile editor Sign On window is displayed. This is an optional parameter.

-L1 *log file*
The name of the error log file to which you want to log informational and error messages. The default is IHSECPED.LOG. If the file is not fully-qualified, it is put in the following directories:

- For Intel: `%BINDIR%\TDS\server\log`
- For UNIX: `$BINDIR/TDS/server/log`

The error log file is continually appended, so that multiple runs of the utility are logged in the same file.

This is an optional parameter.

-V
Forces the utility into verify mode. The response file is processed and compared with the data in the topology server, but no changes are made to the actual topology server database. This is an optional parameter.

-M
Overrides the default add mode of the utility and forces the utility into modify mode. This enables information to be replaced in or deleted from the existing database. This is an optional parameter.

-G
Overrides the default add mode of the utility and forces the utility into modify mode. This enables updates to be made to command sets and profiles by adding to them without replacing existing information. This is an optional parameter.

Return Codes

The command profile editor utility program generates the following return codes:

RC	Explanation
0	Successful. The utility completed successfully. Unless -V was specified in the **cpebatch** command, the database was updated.
4	Warning. The database was updated (unless -V specified) and warning messages were logged.
8	Error. The database was not updated and error messages were logged.
12	Severe Error. The database was not updated and a severe error terminated the program immediately.

dbtransfer

Format

dbtransfer

Purpose

The **dbtransfer** command copies the topology server databases from the default installation directories to the location defined by the *TSERVER_DB* variable. Once the *TSERVER_DB* variable is defined on the topology server workstation, this utility must be run before the topology server is started. This command will not change the contents of the topology server databases in the default installation directories.

getpd

Format

getpd

Purpose

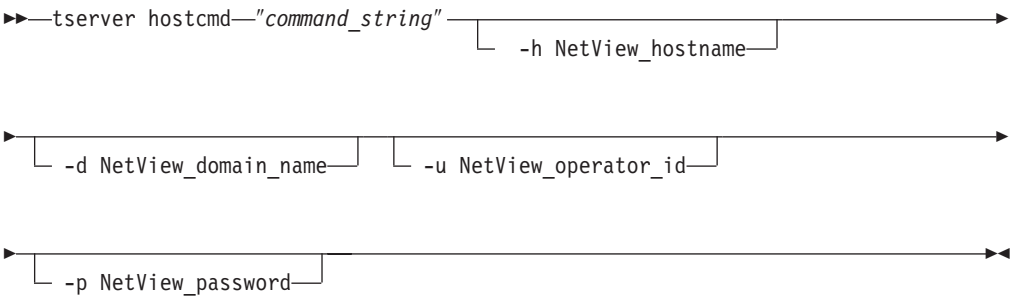
Gathers information about your system environment, error logs, and trace files and stores them into the $BINDIR/TDS/server/tmp/toposerv.xx.tar.Z file. You can send this file to the Tivoli Customer Support to help with problem determination.

This command can be used only on the UNIX platform.

hostcmd

Format

hostcmd



Purpose

Issues commands to the NetView host from a command prompt on the topology server.

Parameters

"command string"
The command to be sent to the NetView host.

-h *NetView_hostname*
The IP address or host name of the NetView host where you want to issue the command.

-d *NetView_domain_name*
NetView domain name where you want to issue the command.

-u Specifies the NetView operator ID where you want to issue the command. This ID overrides any preset NetView operator ID (such as the ID set in the ihsshstc.cfg file or **hostcmdoper** command).

If the `-u` operand is specified without the `-p` operand, you are prompted to enter the NetView password.

-p Specifies the NetView password where you want to issue the command. This password overrides any preset NetView password (such as the password set in the ihsshstc.cfg file or **hostcmdoper** command).

If the `-p` operand is specified without the `-u` operand, you are prompted to enter the NetView operator ID.

Usage

The default is to run the command on the NetView host where the NETCONV session was initiated. See “Establishing Communication Between the NetView Host and the Topology Server” on page 72 for more information on setting up a NETCONV session.

The **hostcmd** command is issued from the command line or a script file. The response to the **hostcmd** command is displayed in the same command window you use to issue the command.

The **hostcmd** command is located in one of the following directories:

- For Intel: `%BINDIR%\TDS\server\bin`
- For UNIX: `$BINDIR/TDS/server/bin`

For UNIX, Windows NT, or Windows 2000, you can either change to this directory before executing the **hostcmd** command, or add the directory path to your PATH environment variable. For OS/2, the directory has already been added to the PATH environment variable.

See “Defining the NetView for OS/390 User ID and Password on the Topology Server” on page 15 for more information about presetting the NetView operator ID and password.

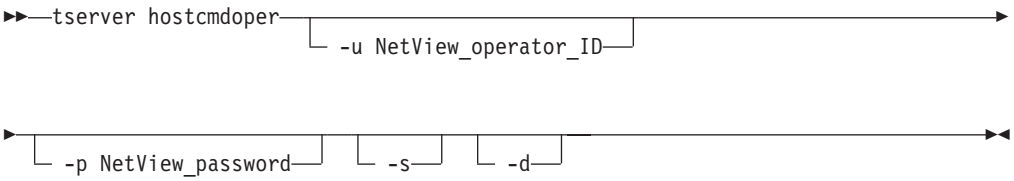
If you want to route command responses to a file or log, or to automate actions based on the command responses, you can customize the command exit source file, `ihsshstc.c`, located in one of the following directories:

- For Intel: `%BINDIR%\TDS\server\sample`
- For UNIX: `$BINDIR/TDS/server/sample`

hostcmdoper

Format

hostcmdoper



Purpose

Issue this command before issuing the **hostcmd** command to set the NetView operator ID and password. The **hostcmdoper** command can also be used to display or delete the NetView operator ID and password in the ihsshostc.cfg file.

Parameters

-u Specifies the NetView operator ID where you want to issue a **hostcmd** command.

-p Specifies the NetView password where you want to issue the **hostcmd** command.

-s Displays the NetView operator ID stored in the ihsshostc.cfg file. This operand cannot be specified with any other operand.

-d Deletes the ihsshostc.cfg file which stores the NetView operator ID and password. This operand cannot be specified with any other operand.

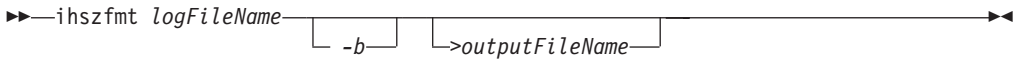
Usage

If the **hostcmdoper** command is never issued or if the -d option is run, and then the **hostcmd** command is run, the **hostcmd** command will either use the operator ID and password stored in the ihsshstc.cfg file (password is not encrypted), or will prompt the user for the operator ID and password.

ihszfmt

Format

ihszfmt



Purpose

Formats the topology server error and trace logs. The output from the command is directed to `stdout`. The error logs and trace files are located in one of the following:

- For Intel: `%BINDIR%\TDS\server\log`
- For UNIX: `$BINDIR/TDS/server/log`

Parameters

logFileName
Specifies the topology server error log or trace file to format. To format the error log, specify either `ihserror.log` or `ihserror.bak`. To format the trace log, specify either `ihstrace.log` or `ihstrace.bak`.

-b Specifies to suppress the formatting of the log in EBCDIC. This parameter does not affect the formatting of the log in ASCII. Formatting of the log in EBCDIC is important because the log contains data being sent between the topology server and the NetView host; thus, it is not recommended to suppress this formatting.

outputFileName
Specifies the file name for the formatted error log or trace file.

ihszset

Format

ihszset

Purpose

Starts the command line interface that enables you to set the trace options for the topology server. If you do not specify an option, **ihszset** starts the GUI interface that enables you to set the trace options for the topology server.

Parameters

-help | -? | -h
Specifies a help menu to be displayed describing all the options you can specify with the **ihszset** command.

ihszsett

Format

ihszsett

Purpose

Starts the graphical user interface, which enables you to set the trace options for the topology server.

service

Format

service

Purpose

Sets up the topology server to run as a Windows NT or Windows 2000 service.

Parameters

account_name

If the service type is SERVICE_WIN32_OWN_PROCESS, this name is the account name in the form of 'DomanName\Username', which the service process logs on as when it runs. If the account belongs to the built-in domain, '.Username' can be specified. Services of type SERVICE_WIN32_SHARE_PROCESS are not a valid specification of an account other than LocalSystem. If NULL is specified, the service logs on as the 'LocalSystem' account, in which case the password parameter must be null.

If the service type is SERVICE_KERNAL_DRIVER or SERVICE_FILE_SYSTEM_DRIVER, this name is the Windows NT driver object name (that is, '\FileSystem\Rdr' or '\Driver\Xns'), which the input and output (I/O) system uses to load the device driver. If NULL is specified, the driver is run with a default object name created by the I/O system, based on the service name.

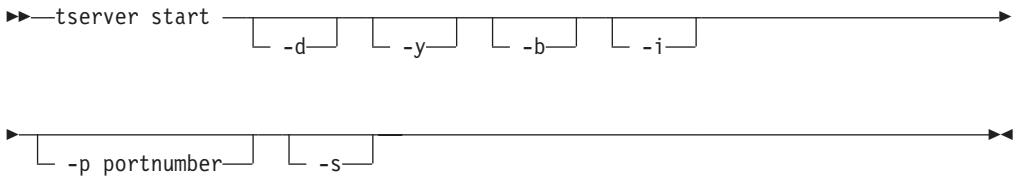
password

Contains the password to the account name specified by the IpServiceStartName parameter, if the service type is SERVICE_WIN32_OWN_PROCESS or SERVICE_WIN32_SHARE_PROCESS. If the pointer is NULL or if it points to an empty string, the service has no password. If the service type is SERVICE_KERNAL_DRIVER or SERVICE_FILE_SYSTEM_DRIVER, this parameter is ignored.

start

Format

start



Purpose

Starts the topology server and communications server, then displays a topology server window and a communications server window. If the topology server is already running either in the background as a service or daemon, in Windows NT, or in Windows 2000, an additional instance of the topology server is not started.

Parameters

-d Specifies that the topology server be run in debug mode. Additional information is written to both the topology server window and to the topology server message log.

-y Generates the default topology server databases if no databases exist. Do not use unless requested by Tivoli Customer Support.

-b Writes additional information to the message log when event data is received. This is helpful if instrumentation is being developed, since it displays the data received along with the information as it is processed.

-i Runs the topology server in IP mode only. Communications through LU 6.2 is not attempted.

-p *portnumber*
Specifies which port will be used for communications with the topology consoles. This does not override what is specified in the services file.

-s Starts the server processes in the background for UNIX. On Windows NT or Windows 2000, this starts the server processes as a Windows NT or Windows 2000 service, assuming the server has been set up to run as a service. This parameter is not used for OS/2.

Note: This parameter must be specified first.

Usage

The topology server and communications server windows only display messages. Use the **utility** command to manually write the topology server databases to disk, dump the server databases, dump the server semaphores, and so on.

stop

Format

stop

Purpose

Stops the topology server processes.

Parameters

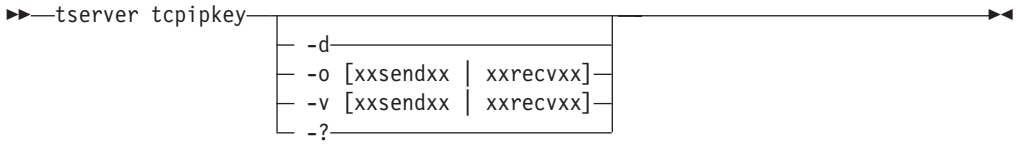
-f Use the force flag if one of the topology server processes ended abnormally or if the topology server is hung. The force flag terminates any remaining topology server processes and then cleans up any remaining interprocess communications (IPC) resources.

Note: This is applicable only to the UNIX platform.

tcpipkey

Format

tcpipkey



Purpose

Enables you to specify the send and receive keys used for encrypting and decrypting data sent or received by the workstation on the IP connection with the NetView host.

Parameters

-d Resets the keys to their default values.

-o Sets the keys used for the NETCONV connection with the NetView host. Enter the keys in the same format as they were entered in `DSITCPRF`. The first key is used to encrypt data sent from the workstation to the NetView host. The second key is used to decrypt data received by the workstation from the NetView host.

xxsendxx
Used to encrypt data sent from the topology server to the NetView host.

xxrecxx
Used to decrypt data received by the topology server.

-v Compares the two keys provided against the two NetView keys that are stored. If the keys match, `Yes` is returned. If the keys do not match, `No` is returned. If only one key is provided on the command line, both keys are prompted. If no keys are provided, the user is prompted for both keys.

xxsendxx
Used to encrypt data sent from the topology server to the NetView host.

xxrecxx
Used to decrypt data received by the topology server.

-? Displays the command syntax.

Usage

Keys must be either eight or sixteen characters long. If a sixteen-character key is entered, it is assumed to be a hexadecimal representation of the key and it is compressed down to eight bytes. If only one key is provided on the command line, then prompts are issued for both keys. If no keys are provided, then the user is prompted for both keys. The keys are stored in an encrypted format in the following file:

- For Intel: `%BINDIR%\TDS\server\config\ihssckey.cfg`
- For UNIX: `$BINDIR/TDS/server/config/ihssckey.cfg`

tserver

Format

tserver

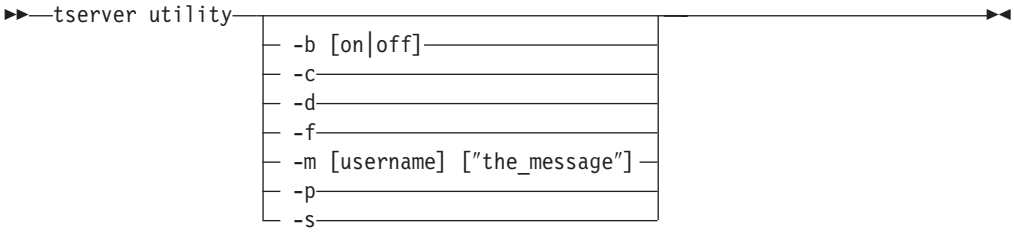
Purpose

Use **tserver** before some of the topology server commands (for example, **tserver stop**). The commands that require the **tserver** prefix are documented as such in this appendix. On the Intel platform, the topology server commands are a combination of command files and shell scripts. The **tserver** command ensures that the appropriate environment is set up and then invokes the requested command.

utility

Format

utility



Purpose

Enables you to manually write the topology server databases to disk, dump the server databases, dump the server semaphores, or send a message to topology consoles signed on to the same topology server.

Parameters

-b [on|off]
Turns the instrumentation-related topology server message logging on or off.

-c Manually writes (check points) the topology server databases to disk.

-d Dumps the topology server databases to the following directory:

- For Intel: `%BINDIR%\TDS\server\log`
- For UNIX: `$BINDIR/TDS/server/log`

-f Dumps the topology server databases, without semaphore access protection, to the following directory:

- For Intel: `%BINDIR%\TDS\server\log`
- For UNIX: `$BINDIR/TDS/server/log`

-m Sends messages to topology consoles connected to the topology server.

user_name
The user name of the topology console to whom you want to send the message, or use **all** to broadcast the message to all topology consoles signed on to the same server.

"**the_message**"
The message to send to the topology console.

Note: The double quotations are required unless the message is a single token.

-p Displays the current settings of the server properties file.

-s Dumps the server semaphores to the screen and to the message log.

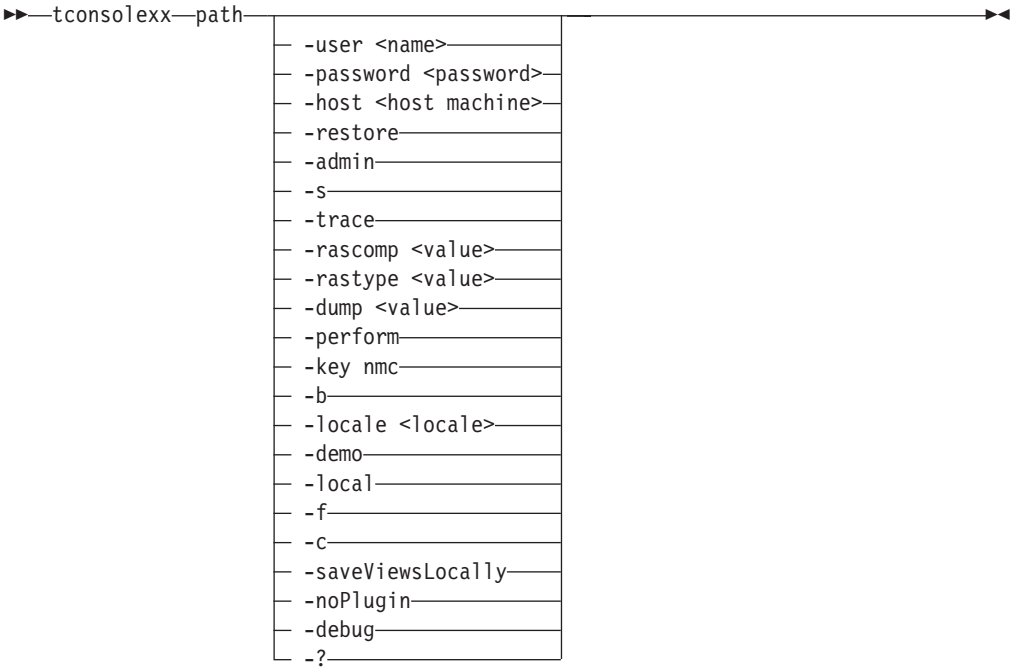
Appendix B. Topology Console Commands

The topology console commands provided in this appendix are intended to be used as reference material. The format in the following commands is in the form of syntax diagrams.

tconsolexx

Format

tconsolexx



Purpose

Starts the topology console from the command line. It is recommended that you start this directly from the topology console bin directory. Following are the possible values for xx:

95 Windows 95 or Windows 98

NT Windows NT and Windows 2000

OS2 OS/2

.sh UNIX

Parameters

path
Specifies the top level of the topology console installation path:

- For Intel: `%BINDIR%\..\generic_unix\TDS\client`
- For UNIX: `$BINDIR/../generic_unix/TDS/client`

-user <user>
Specifies the topology console sign on user ID. See “Using the Topology Console Sign On Window” on page 74 for more information.

-password <password>
Specifies the topology console sign on password. See “Using the Topology Console Sign On Window” on page 74 for more information.

-host <host machine>
Specifies the topology console sign on host name and possible port number. See “Using the Topology Console Sign On Window” on page 74 for more information.

-restore
Specifies the topology console sign on restore console attribute. See “Using the Topology Console Sign On Window” on page 74 for more information.

-admin
Specifies the topology console sign on administrator attribute. See “Using the Topology Console Sign On Window” on page 74 for more information.

-s Automatically signs on using the specified options, previously saved values, or both. See “Using the Topology Console Sign On Window” on page 74 for more information.

-trace
This option is for Tivoli Customer Support only. It enables default tracing.

-rascomp <value>
This option is for Tivoli Customer Support only. The <value> variable can be obtained from the Service page of the Console Properties notebook.

-rastype <value>
This option is for Tivoli Customer Support only. The <value> variable can be obtained from the Service page of the Console Properties notebook.

-dump <value>
This option is for Tivoli Customer Support only.

-perform
This option is for Tivoli Customer Support only. Enables performance tracing.

-key nmc
Optional keyword.

-b Use buffered tracing.

-locale
Override the default locale. The format for *locale* follows:

```
langCode [countryCode]
```

-demo
Starts the topology console disconnected from the topology server.

-local
Starts the topology console disconnected from the topology server.

-f Suppresses automatic synchronization of support files from the topology server.

-c Suppresses automatic synchronization of topology console code from the topology server. This is not recommended.

-saveViewsLocally
When Save View Customization is used while there is a connection to a topology server, the view is saved in a stand-alone file on the topology console workstation. This option is used only for capturing live views for use in a demo.

-noPlugin
Suppresses loading of any plug-ins.

-debug
Enables built-in debugging code. This option is for Tivoli Customer Support only.

-? Display the command line usage.

Usage

All scripts support the following optional environment variables:

TCONSOLE_BACKDOOR
Java code library (or libraries) placed at front of CLASSPATH. This option is for Tivoli Customer Support only.

TCONSOLE_CLASSPATH
Java code library (or libraries) placed at end of CLASSPATH. Typically, this is used to define other Java classes to be accessed by the NetView management console, such as the NMC-3270.

TCONSOLE_JAVAOPTS
Command line argument(s) for Java program. This option is for Tivoli Customer Support only.

tappxx

Format

tappxx

Purpose

Starts the topology console utility functions. It is recommended that you start this directly from the topology console bin directory. Following are the possible values for xx:

95 Windows 95 or Windows 98

NT Windows NT and Windows 2000

OS2 OS/2

.sh UNIX

Parameters

path
Specifies the top level of the topology console installation path:

- For Intel: `%BINDIR%\..\generic_unix\TDS\client`
- For UNIX: `$BINDIR/../generic_unix/TDS/client`

class_name
Specifies the class name for the topology console provided utility.

args
Specifies any arguments that are required for the utility.

Usage

All scripts support the following optional environment variables:

TCONSOLE_BACKDOOR
Java code library (or libraries) placed at front of CLASSPATH. This option is for Tivoli Customer Support only.

TCONSOLE_CLASSPATH
Java code library (or libraries) placed at end of CLASSPATH. Typically, this is used to define other Java classes to be accessed by NetView management console, such as the NMC-3270.

TCONSOLE_JAVAOPTS
Command line arg(s) for Java program. This option is for Tivoli Customer Support only.

Appendix C. Sending Commands to Multiple NetView for OS/390 Domains

When sending certain commands from the topology console, such as **Activate**, **Inactivate**, or **Recycle**, to the NetView host, there might be more than one NetView domain to run the command against. This occurs only when you right click a Systems Network Architecture topology manager (SNATM) resource with a command indicator value of 32769 and the command is to be run at NetView for OS/390 with the IHSXTHCE command exit. For example, when you select an SNATM resource to issue a command against, more than one NetView domain might be monitoring that resource. In this case, a multiple domain dialog is displayed, so you can select one or more NetView domains. Following is an example of the multiple domain dialog.

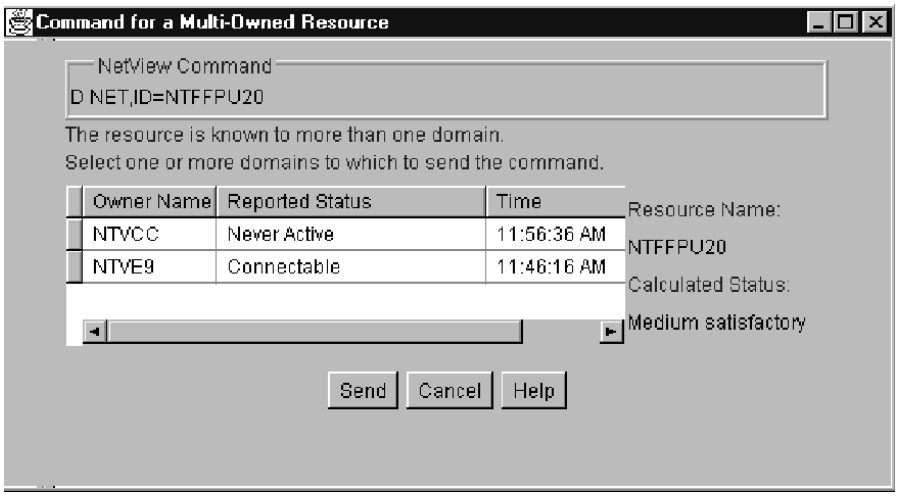


Figure 29. Multiple Domain Dialog Example

You can also specify one or more NetView domains as a default. In this case, when a command is issued against a resource, only the NetView domains specified as the default receive the command.

To specify a NetView domain as the default, follow these steps:

1. Create one or more files named:
   - **userMultiDom.properties** where user is the user name of an operator. Use this file if you want to specify one or more NetView domains as a default for a specific operator.
   - **DefaultMultiDom.properties** to specify one or more NetView domains as the default for all operators.

   Following is an example of these properties:

   ```
   autoSelect.1 = NTA09
   autoSelect.2 = NTM12
   autoSelect.3 = NTM13
   autoSend = true
   ```

   When `autoSend = true`, commands are executed on the NetView domains specified in the properties file without any operator interaction (the multiple

domain dialog is not displayed). All other NetView domains are ignored. A value of `autoSend = false` indicates that the multiple domain dialog should be displayed for the operator with the domains specified in the properties file automatically selected.

2. Store these files on the topology server workstation in one of the following directories:
   - For Intel: `%BINDIR%\TDS\server\db\current\settings`
   - For UNIX: `$BINDIR/TDS/server/db/current/settings`

Appendix D. Automatic File Download at Console Log On

This appendix describes how files are downloaded from the server to the console when the console signs on to the server.

The installation of the NMC console occurs in the following two phases:

1. During the installation process
2. The first time the console logs on to the server

During Installation

During the installation process, the smallest possible set of files is installed on the local workstation. This includes the following types of files:

- NMC console code
- MRI (readable strings) for the appropriate language
- A subset of background, help, and icon files

Note: The exception to the rule of installing the smallest possible set of files on the workstation during installation occurs when a custom install is performed and the Productivity Kit is selected. This installs all files necessary to run the NMC Console in demo mode (not requiring a NetView management console Server).

During Initial Sign On

The first time the console signs on to the server following installation, files are automatically downloaded from the following server directories:

- For Intel: %BINDIR%/TDS/server/db/current/
- For UNIX: $BINDIR$\TDS\server\db\current

These directories and their descriptions follow:

backgrounds
View background maps/images

bin NMC console binary support files including scripts

icons Resource and company icons

help Online help support files including:
- NetView management console product specific help files
- Customer Java application help file(s)

lib Java code including the following files:
- NMC console Java code (ihseuc.jar)
- NetView management console product personality file (nmc.properties)
- Customer Java application and/or plug-in JAR files

During Subsequent Sign On

The files downloaded during the initial sign on are automatically checked for updates each time the console successfully signs onto the server. That is, when the console downloads a file, the console stores the time stamp of the file on the server workstation. If the time stamp has changed, the file is downloaded again.

If the console connects to the same server, these files are updated only when changes have been made. However, if the console connects to different servers on different platforms, the time stamps across servers will be close, but not identical. Therefore, a value in the defaultscheme.properties file can be set to enable you to control exactly when the download occurs. This control applies when reconnecting to the same server or to different servers. See “Customizing the Automatic Download of Files At Log On” on page 29 for more detailed information.

The `lib` directory is handled differently than the others. Files installed by the Console, and named in the contents.properites file in the lib directory, are only downloaded if a different build of the Console is placed in this directory. That is, the time stamps of these files are not cross-checked. The time stamp cross-check does occur for files in the lib directory which were provided by the customer. However, the defaultscheme.properties file has not yet been read in; therefore control over when these files are downloaded cannot be customized. See the defaultscheme.properties file for more information and for the internal value that the Console uses for the time stamp cross-check.

When a file in the `lib` directory needs to be downloaded, message `IHS1137` is displayed and the file must be downloaded before sign on can continue. These files might need to be reloaded into memory. For files in all other directories, the file is downloaded and sign on processing continues automatically.

See “Appendix B. Topology Console Commands” on page 161 for more information about command line arguments to suppress this downloading process.

Index

Special Characters

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

R

S

T

U

V

W

File Number: S370/4300/30xx-50
Program Number: 5697–B82

Printed in the United States of America
on recycled paper containing 10%
recovered post-consumer fiber.

GC31-8665-02