NetView Management Console User’s Guide

Version 5 Release 1
Tivoli NetView for z/OS NetView Management Console User’s Guide

Copyright Notice

© Copyright IBM Corporation 1997, 2002. All rights reserved. May only be used pursuant to a Tivoli Systems Software License Agreement, an IBM Software License Agreement, or Addendum for Tivoli Products to IBM Customer or License Agreement. No part of this publication may be reproduced, transmitted, transcribed, stored in a retrieval system, or translated into any computer language, in any form or by any means, electronic, mechanical, magnetic, optical, chemical, manual, or otherwise, without prior written permission of IBM Corporation. IBM Corporation grants you limited permission to make hardcopy or other reproductions of any machine-readable documentation for your own use, provided that each such reproduction shall carry the IBM Corporation copyright notice. No other rights under copyright are granted without prior written permission of IBM Corporation. The document is not intended for production and is furnished “as is” without warranty of any kind. All warranties on this document are hereby disclaimed, including the warranties of merchantability and fitness for a particular purpose.

U.S. Government Users Restricted Rights—Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corporation.

Trademarks

IBM, the IBM logo, Tivoli, the Tivoli logo, AIX, APPN, BookManager, DB2, NetView, OS/390, Tivoli Enterprise Console, TME, VisualAge, VTAM, and z/OS are trademarks or registered trademarks of International Business Machines Corporation 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 and logos 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 z/OS.
## Contents

**Preface** ........................................... ix
Who Should Read This Document ........................................... ix
What This Document Contains ........................................... ix
Publications .................................................. x
  Prerequisite and Related Documents .................................. x
  Accessing Publications Online ....................................... x
  Ordering Publications ........................................... xi
  Providing Feedback about Publications ................................ xi
Contacting Customer Support .............................................. xi
Accessibility Information .................................................. xi
  Keyboard Access .................................................. xii
Conventions Used in This Document ....................................... xii
Platform-specific Information ............................................. xii
Terminology .......................................................... xii
Reading Syntax Diagrams .................................................. xiii
  Required Syntax ................................................... xiv
  Optional Keywords and Variables ..................................... xiv
  Default Values ................................................... xiv
  Long Syntax Diagrams ................................................ xv
  Syntax Fragments ................................................... xv
  Commas and Parentheses ............................................. xvi
  Highlighting, Brackets, and Braces .................................... xvii
  Abbreviations .................................................... xviii

**Part 1. Overview** ............................................ 1

**Chapter 1. Introduction to the NetView Management Console.** .................. 3
What Can You Do with NetView Management Console? .............................. 3
How Does NetView Management Console Work? ..................................... 3
Topology Server ........................................................................ 4
Topology Console ........................................................................ 4
Real and Aggregate Resources ..................................................... 4

**Chapter 2. Understanding Views** ............................................. 7
RODM-Based Views ....................................................................... 7
Network Views ............................................................................ 7
Exception Views ........................................................................... 8
Configuration Views ..................................................................... 8
More Detail Views ......................................................................... 11
Locate Failing Resources ........................................................... 12
Customized Views ......................................................................... 12
Views Containing Resources for Which You Are Not Authorized ................. 13
Views Containing Scheduled Resources ........................................... 13
Displaying Views in a Web Browser ................................................ 13

**Part 2. Installing and Customizing the NetView Management Console.** ........ 15

**Chapter 3. Installing the NetView Management Console.** ......................... 17
Installing the Topology Console ................................................... 17
Installing the Topology Server ..................................................... 17
Defining the NetView for z/OS User ID and Password on the Topology Server .................................................. 17

**Chapter 4. Customizing the NetView Management Console Topology Server.** ...... 19
### Chapter 5. Customizing the NetView Management Console Topology Console

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Customizing Topology Console Features</td>
<td>25</td>
</tr>
<tr>
<td>Adding and Customizing Topology Console Icons</td>
<td>25</td>
</tr>
<tr>
<td>Adding and Customizing Topology Console Backgrounds</td>
<td>26</td>
</tr>
<tr>
<td>Displaying Customized Help</td>
<td>26</td>
</tr>
<tr>
<td>Configuring a Web Browser to Display Views</td>
<td>27</td>
</tr>
<tr>
<td>Advanced Topology Console Customization</td>
<td>28</td>
</tr>
<tr>
<td>Customizing Your Online Help Facility</td>
<td>28</td>
</tr>
<tr>
<td>Enabling User Flags</td>
<td>29</td>
</tr>
<tr>
<td>Adding a Flag to the Context Menu</td>
<td>31</td>
</tr>
<tr>
<td>Running a Console Class</td>
<td>35</td>
</tr>
<tr>
<td>Customizing Web Server Enablement</td>
<td>35</td>
</tr>
<tr>
<td>Customizing the View Bar Layout</td>
<td>36</td>
</tr>
<tr>
<td>Customizing the Automatic Download of Files At Log On</td>
<td>37</td>
</tr>
<tr>
<td>Overriding the Default Date/Time Format</td>
<td>38</td>
</tr>
<tr>
<td>Customizing Line Thickness</td>
<td>41</td>
</tr>
</tbody>
</table>

### Chapter 6. Creating a Demo

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Capturing Live Views from your NetView Management Console System</td>
<td>43</td>
</tr>
<tr>
<td>Using Basic Data Files</td>
<td>45</td>
</tr>
<tr>
<td>Integrating Captured Views into the Demo</td>
<td>50</td>
</tr>
<tr>
<td>Updating the Business Tree</td>
<td>50</td>
</tr>
<tr>
<td>Renaming Navigation Views</td>
<td>52</td>
</tr>
<tr>
<td>Defining New Resource Types in Saved Views</td>
<td>53</td>
</tr>
<tr>
<td>Finding the Resource ID</td>
<td>54</td>
</tr>
<tr>
<td>Defining a Node Resource in a View</td>
<td>54</td>
</tr>
<tr>
<td>Defining a Link Resource in a View</td>
<td>55</td>
</tr>
<tr>
<td>Defining View Information</td>
<td>56</td>
</tr>
<tr>
<td>Defining a Demo View</td>
<td>57</td>
</tr>
</tbody>
</table>

### Chapter 7. Topology Console Java Applications and Plug-ins

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supplied Support Files</td>
<td>59</td>
</tr>
<tr>
<td>Installing the Examples</td>
<td>59</td>
</tr>
<tr>
<td>Enabling the Examples</td>
<td>60</td>
</tr>
<tr>
<td>Compiling the Examples</td>
<td>60</td>
</tr>
<tr>
<td>Tracing the Examples</td>
<td>61</td>
</tr>
<tr>
<td>Problem Determination</td>
<td>61</td>
</tr>
<tr>
<td>Java Applications</td>
<td>61</td>
</tr>
<tr>
<td>Java Application Examples</td>
<td>62</td>
</tr>
<tr>
<td>Java Application Development Process</td>
<td>62</td>
</tr>
<tr>
<td>Defining the Example Java Applications</td>
<td>63</td>
</tr>
<tr>
<td>Running the Example Java Applications</td>
<td>63</td>
</tr>
<tr>
<td>Java Plug-Ins</td>
<td>64</td>
</tr>
<tr>
<td>Supported Plug-Ins</td>
<td>64</td>
</tr>
<tr>
<td>Plug-In Definitions File</td>
<td>66</td>
</tr>
<tr>
<td>Plug-In Examples</td>
<td>66</td>
</tr>
<tr>
<td>Plug-In Development Process</td>
<td>67</td>
</tr>
<tr>
<td>Defining the Example Java Plug-Ins</td>
<td>67</td>
</tr>
<tr>
<td>Running the Example Java Plug-Ins</td>
<td>67</td>
</tr>
</tbody>
</table>
### Chapter 8. Configuring Property Files for Locally Launched Applications
- Defining the Pop-up Menu Items .................................................. 69
- Response File Input ....................................................................... 69
- Creating a Response File for Browser .......................................... 70
- Defining the Properties File ......................................................... 71

### Part 3. Using NetView Management Console ................................. 77

### Chapter 9. Operating the NetView Management Console .............. 79
- Starting the Topology Server ....................................................... 79
  - Starting the Topology Server from the Desktop Icon .................. 79
  - Manually Starting the Topology Server ..................................... 79
  - Starting the Topology Server as a Windows Service .................. 80
  - Starting the Topology Server as a Daemon ............................... 80
  - Establishing Communication Between the NetView Host and the Topology Server ................................................................. 80
- Starting the Topology Console .................................................... 81
  - Selecting the Desktop Icon in Windows ..................................... 81
  - Using a Line Command ............................................................. 81
  - Using the Topology Console Sign On Window ........................... 81
- The Topology Console Window .................................................... 83
  - The View Area .......................................................................... 85
  - The Filter Bar .......................................................................... 86
- NetView Management Console Online Help ................................... 86
- NetView Management Console Features and Functions ................. 86
  - Issuing IP Commands ................................................................ 86
  - The NetView Resource Manager (NRM) ..................................... 87
  - Using the RODM Collection Manager With NetView Management Console ............................................................... 88
- NetView Management Console Topology Server Databases .......... 89
  - Writing Server Information to the Topology Server Databases .... 89
  - When the Topology Server Databases Are Corrupted ................. 89
  - Creating and Restoring a Permanent Copy of the Topology Server Databases ................................................................. 90
- Stopping the Topology Console .................................................... 90
- Stopping the Topology Server ...................................................... 90
  - With the Service Version on Windows ...................................... 90
  - Using a Line Mode Command ..................................................... 91

### Chapter 10. Using the NetView Management Console Command Profile Editor ...................................................... 93
- Understanding the Command Profile Editor .................................. 93
  - Resource Manager Objects ....................................................... 93
  - Commands and Command Set Objects ....................................... 93
  - Profile Objects .......................................................................... 94
  - Operator Objects ...................................................................... 94
- Using the Command Profile Editor Batch Utility ............................ 95
  - Starting the Command Profile Batch Utility ............................... 96
  - Input and Output Files of the Response File ............................... 96

### Chapter 11. Using the Topology Server Command Exits .................. 105
- Command Profiles ...................................................................... 105
- Understanding Topology Server Command Exits .......................... 105
- Using Topology Server Command Exits ....................................... 106
  - IHSXTHCE Command Exit ........................................................ 107
  - IHSXTJAM Command Exit ........................................................ 108
  - IHSXTHCE Command Exit ........................................................ 108
  - Substitution Variables ................................................................ 108
- Writing Topology Server Command Exits ..................................... 110
  - Available Programming Languages ......................................... 111
  - Command Exit Flow Scenario .................................................. 111
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Writing C Exits</td>
<td>112</td>
</tr>
<tr>
<td>Building C Command Exit Programs</td>
<td>112</td>
</tr>
<tr>
<td>Installing C Programs as Command Exits</td>
<td>113</td>
</tr>
<tr>
<td>C Parameter Block (EGVE_PARAMETERS32 Structure)</td>
<td>114</td>
</tr>
<tr>
<td>Command Exit Functions for C</td>
<td>123</td>
</tr>
<tr>
<td>Return Codes</td>
<td>126</td>
</tr>
<tr>
<td>Invoking Command Exits with a C Interface</td>
<td>129</td>
</tr>
<tr>
<td>IHSDDNATV Command Exit</td>
<td>129</td>
</tr>
<tr>
<td>IHSXTHCE Command Exit</td>
<td>130</td>
</tr>
<tr>
<td>IHSXTJAV Command Exit</td>
<td>132</td>
</tr>
<tr>
<td>IHSXTJCR Command Exit</td>
<td>133</td>
</tr>
<tr>
<td>Determining Additional Resource Information</td>
<td>134</td>
</tr>
<tr>
<td>Resource Manager Determination</td>
<td>134</td>
</tr>
<tr>
<td>Real or Aggregate Determination</td>
<td>134</td>
</tr>
<tr>
<td>Part 4. Appendixes</td>
<td>137</td>
</tr>
<tr>
<td>Appendix A. Topology Server Commands</td>
<td>139</td>
</tr>
<tr>
<td>config</td>
<td>140</td>
</tr>
<tr>
<td>cpebatch</td>
<td>141</td>
</tr>
<tr>
<td>dbtransfer</td>
<td>143</td>
</tr>
<tr>
<td>getpd</td>
<td>144</td>
</tr>
<tr>
<td>hostcmd</td>
<td>145</td>
</tr>
<tr>
<td>hostcmdoper</td>
<td>147</td>
</tr>
<tr>
<td>ihsfmt</td>
<td>148</td>
</tr>
<tr>
<td>ihsset.</td>
<td>149</td>
</tr>
<tr>
<td>ihszett</td>
<td>150</td>
</tr>
<tr>
<td>service.</td>
<td>151</td>
</tr>
<tr>
<td>start</td>
<td>152</td>
</tr>
<tr>
<td>stop</td>
<td>153</td>
</tr>
<tr>
<td>tcpipkey</td>
<td>154</td>
</tr>
<tr>
<td>tserver</td>
<td>155</td>
</tr>
<tr>
<td>utility</td>
<td>156</td>
</tr>
<tr>
<td>Appendix B. Topology Console Commands</td>
<td>157</td>
</tr>
<tr>
<td>tconsolexx</td>
<td>158</td>
</tr>
<tr>
<td>tappxx</td>
<td>161</td>
</tr>
<tr>
<td>Appendix C. Launching and Using the NetView Management Console from Other Applications</td>
<td>163</td>
</tr>
<tr>
<td>Servlets</td>
<td>163</td>
</tr>
<tr>
<td>Setup</td>
<td>163</td>
</tr>
<tr>
<td>Using the Servlets</td>
<td>164</td>
</tr>
<tr>
<td>IhsLocRes Servlet</td>
<td>165</td>
</tr>
<tr>
<td>IhsRunning Servlet</td>
<td>166</td>
</tr>
<tr>
<td>locRes Java Class</td>
<td>167</td>
</tr>
<tr>
<td>nmcRunning Java Class</td>
<td>168</td>
</tr>
<tr>
<td>tlocResxx Script</td>
<td>169</td>
</tr>
<tr>
<td>Appendix D. Sending Commands to Multiple NetView Domains</td>
<td>171</td>
</tr>
<tr>
<td>Appendix E. Auditing and the ihsaudit.xml File</td>
<td>173</td>
</tr>
<tr>
<td>Elements in the ihsaudit.xml file</td>
<td>173</td>
</tr>
<tr>
<td>action ELEMENT</td>
<td>174</td>
</tr>
<tr>
<td>admin ELEMENT</td>
<td>174</td>
</tr>
<tr>
<td>aggPri ELEMENT</td>
<td>174</td>
</tr>
<tr>
<td>aggregation ELEMENT</td>
<td>174</td>
</tr>
</tbody>
</table>
Appendix F. Automatic File Download at Console Log On .................................................. 197

During Installation ........................................................................................................ 197
During Initial Sign On ................................................................................................. 197
During Subsequent Sign On ........................................................................................ 198

Appendix G. Converting NGMF Command Sets .......................................................... 199
Preface

This document provides overview information for the NetView® management console (NMC) interface to Tivoli® NetView for z/OS™. 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 interface to Tivoli NetView for z/OS.

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 7 describes some basic information about NetView management console views.
- "Chapter 3. Installing the NetView Management Console" on page 17 contains information about installing the NetView management console.
- "Chapter 4. Customizing the NetView Management Console Topology Server" on page 19 contains information about customizing the NMC topology server.
- "Chapter 5. Customizing the NetView Management Console Topology Console" on page 25 contains information about customizing the NMC topology console.
- "Chapter 6. Creating a Demo" on page 43 describes how to create NetView management console demos.
- "Chapter 8. Configuring Property Files for Locally Launched Applications" on page 69 describes the NetView management console web launch Java application.
- "Chapter 9. Operating the NetView Management Console" on page 79 contains information about operating the NetView management console, including starting and stopping the topology server and topology console: as well as detailed information about the NetView management console window.
- "Chapter 10. Using the NetView Management Console Command Profile Editor" on page 93 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 11. Using the Topology Server Command Exits" on page 103 describes how to write command exits to use with NetView management console.
- "Appendix A. Topology Server Commands" on page 139 contains a list of topology server commands.
- "Appendix B. Topology Console Commands" on page 157 contains a list of topology console commands.
• “Appendix C. Launching and Using the NetView Management Console from Other Applications” on page 163 describes using servlets, command-line programs, and scripts to launch the NetView management console from other applications.

• “Appendix D. Sending Commands to Multiple NetView Domains” on page 171 describes how to run a command against one or more NetView domains.

• “Appendix E. Auditing and the ihsaudit_xml File” on page 173 describes the auditing function and the ihsaudit files.

• “Appendix F. Automatic File Download at Console Log On” on page 197 describes how files are downloaded from the server to the console when the console signs on to the server.

• “Appendix G. Converting NGMF Command Sets” on page 199 describes how to convert a command set used with the NetView Graphic Monitor Facility (NGMF) to a command profile editor response file used with NetView management console.

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

Publications

This section lists prerequisite and related documents. It also describes how to access Tivoli publications online, how to order Tivoli publications, and how to make comments on Tivoli publications.

Prerequisite and Related Documents

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

You can find additional product information on these Internet sites:

<table>
<thead>
<tr>
<th>Table 1. Resource Web sites</th>
</tr>
</thead>
<tbody>
<tr>
<td>IBM®</td>
</tr>
<tr>
<td>Tivoli Systems</td>
</tr>
<tr>
<td>Tivoli NetView for z/OS</td>
</tr>
</tbody>
</table>

The Tivoli NetView for z/OS Web site offers demonstrations of the NetView product, 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 the NetView program using the program-to-program interface (PPI)
• Sending and receiving MVS™ commands using the PPI
• Sending TSO commands and receiving responses

Accessing Publications Online

You can access many Tivoli publications online using the Tivoli Information Center, which is available on the Tivoli Customer Support Web site:
These publications are available in PDF format. Translated documents are also available for some products.

**Ordering Publications**

You can order many Tivoli publications online at the following Web site:


You can also order by telephone by calling one of these numbers:

- In the United States: 800-879-2755
- In Canada: 800-426-4968
- In other countries, for a list of telephone numbers, see the following Web site: [http://www.tivoli.com/inside/store/lit_order.html](http://www.tivoli.com/inside/store/lit_order.html)

**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 an e-mail to pubs@tivoli.com.
- Complete our customer feedback survey at the following Web site: [http://www.tivoli.com/support/survey/](http://www.tivoli.com/support/survey/)

**Contacting Customer Support**

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


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

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

**Note:** Additional support for Tivoli NetView for z/OS is available at the NetView for z/OS Web site:

[http://www.tivoli.com/nv390](http://www.tivoli.com/nv390)

Under Related Documents, select Other Online Sources.

The page displayed contains a list of newsgroups, forums, and bulletin boards.

**Accessibility Information**

Refer to *Tivoli NetView for z/OS User’s Guide* for information about accessibility.
Keyboard Access

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

Refer to *Tivoli NetView for z/OS User’s Guide* for more information about keyboard access.

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 z/OS 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 z/OS Licensed Program Specification*.

Terminology

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

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

**NetView**

- Tivoli NetView for z/OS Version 5 Release 1
- 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**  OS/390, or z/OS operating systems.

**RACF®**  RACF is a component of the SecureWay® Security Server for z/OS and OS/390, providing the functions of authentication and access control for OS/390 and z/OS resources and data, including the ability to control access to DB2® objects using RACF profiles. Refer to: [http://www-1.ibm.com/servers/eserver/zseries/zos/security/racfss.html](http://www-1.ibm.com/servers/eserver/zseries/zos/security/racfss.html)

**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 are included in the IBM Communications Server element of the OS/390 and z/OS operating systems. Refer to [http://www.ibm.com/software/network/commserver/about/](http://www.ibm.com/software/network/commserver/about/).

| Table 2 defines the Windows® and UNIX® terms as they are used with the NMC topology console and NMC topology server. |

**Table 2. NMC topology server/NMC topology console Platforms**

<table>
<thead>
<tr>
<th>Term</th>
<th>NMC topology console</th>
<th>NMC topology server</th>
</tr>
</thead>
</table>
| Windows | • Windows NT®  
|         | • Windows 2000  
|         | • Windows NT  
|         | • Windows 2000  |
| UNIX   | • AIX®  
|        | • Solaris  
|        | • HP/UX  
|        | • Red Hat Linux (Intel)  
|        | • SuSE Linux Professional  
|        | • AIX  
|        | • SuSE Linux for S/390®  |

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**

<table>
<thead>
<tr>
<th>Element Position</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>On the command line</td>
<td>Required</td>
</tr>
<tr>
<td>Above the command line</td>
<td>Default</td>
</tr>
<tr>
<td>Below the command line</td>
<td>Optional</td>
</tr>
</tbody>
</table>
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

\[\text{CCPLOADF resname}\]

*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

\[\text{TRANSMSG MEMBER=membername}\]

*Figure 2. Syntax for Variables*

Optional Keywords and Variables

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

DISPREG

\[\text{DISPREG ID=resname}\]

*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 on page xvi 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.
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 xvii for a syntax with the fragments ReMote and FromTo.
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 xvii, the OP operand, for example, contains commas to indicate that you can specify multiple values for the testop variable.
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 xv.

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

<table>
<thead>
<tr>
<th>This element...</th>
<th>Looks like this...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Keyword</td>
<td>CCPLOADF</td>
</tr>
<tr>
<td>Variable</td>
<td>resname</td>
</tr>
<tr>
<td>Operand</td>
<td>MEMBER=membername</td>
</tr>
<tr>
<td>Default</td>
<td>today or INCL</td>
</tr>
</tbody>
</table>
Preface

Abbreviations

Command and keyword abbreviations are described in synonym tables after each command description.
Part 1. Overview

Chapter 1. Introduction to the NetView Management Console

What Can You Do with NetView Management Console? ............................................... 3
How Does NetView Management Console Work? ...................................................... 3

Topology Server ........................................................................................................... 4
Topology Console ......................................................................................................... 4
Real and Aggregate Resources ..................................................................................... 4

Chapter 2. Understanding Views

RODM-Based Views .................................................................................................... 7
Network Views ............................................................................................................... 7
Exception Views ........................................................................................................... 8
Configuration Views ..................................................................................................... 8
  Configuration Parents View ....................................................................................... 8
  Configuration Children View ................................................................................... 8
  Configuration Peers View ....................................................................................... 9
  Configuration Logical View ..................................................................................... 9
  Configuration Physical View .................................................................................... 10
  Configuration Logical and Physical ....................................................................... 10
  Configuration Backbone View ............................................................................... 11
More Detail Views ........................................................................................................ 11
Locate Failing Resources ......................................................................................... 12
Customized Views ....................................................................................................... 12
Views Containing Resources for Which You Are Not Authorized ......................... 13
Views Containing Scheduled Resources ................................................................. 13
Displaying Views in a Web Browser ...................................................................... 13
Chapter 1. Introduction to the NetView Management Console

The NetView management console (NMC) of Enterprise NetView for z/OS V5R1 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 or status changes received for a selected resource.
• Issue predefined commands from context menus, or issue your own commands.
• Stop and restart selected resources.
• Monitor and manage multiple NetView programs.
• Cycle through open views automatically, at intervals.
• Build custom view and aggregate resource collections.
• Monitor resources by exception, to show on the screen only when the resources need the attention of the operator.

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 6. Creating a Demo” on page 43.

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 z/OS. 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 9. Operating the NetView Management Console" on page 79 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 9. Operating the NetView Management Console" on page 79 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.

![Network View Diagram](image_url)
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 shows an exception view.

![Figure 8. Exception View](image)

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](image)

Configuration Children View

Figure 10 on page 9 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 10 shows a view containing resources in the network that are arranged in a configuration based on a logical relationship between resources.
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.

Configuration Logical and Physical

Figure 14 on page 11 shows a view containing resources in the network that are arranged in a configuration based on a logical and physical relationship between resources.
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.

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 12 shows the contents of a resource, or more details about the resource.
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.

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 z/OS 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 z/OS 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 topology console has a scheduled system status. The resource continues to receive system status updates in RODM but they are not sent to the NMC topology 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 topology 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.

<table>
<thead>
<tr>
<th>If you want information about...</th>
<th>Refer to...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Displaying views in a Web browser</td>
<td>&quot;Configuring a Web Browser to Display Views&quot; on page 27</td>
</tr>
</tbody>
</table>
Part 2. Installing and Customizing the NetView Management Console

Chapter 3. Installing the NetView Management Console
Installing the Topology Console ................................................................. 17
Installing the Topology Server ................................................................. 17
Defining the NetView for z/OS User ID and Password on the Topology Server. 17

Chapter 4. Customizing the NetView Management Console Topology Server
Customizing Topology Server Properties File .................................................. 19
Customizing the Functions to be Audited ..................................................... 19
Customizing the Audit Log for Viewing ...................................................... 20
Customizing the Size of the Audit Log ......................................................... 22
Customizing the Length of Command Responses within the Audit Log .......... 22
Customizing Special Characters to be Browsed .......................................... 23
Customizing the Replacement Character for Carriage Returns and Line Feeds . 23

Chapter 5. Customizing the NetView Management Console Topology Console
Customizing Topology Console Features ..................................................... 25
Adding and Customizing Topology Console Icons ....................................... 25
Adding and Customizing Topology Console Backgrounds ............................. 26
Configuring a Web Browser to Display Views . ................................................ 27
Designating a Console as a Web Server ....................................................... 27
Designating Multiple Consoles as Web Servers .......................................... 27
Using the Web Browser ................................................................................. 27
Logging Web Server Messages ..................................................................... 28
Advanced Topology Console Customization .................................................. 28
Customizing Your Online Help Facility ....................................................... 28
Enabling User Flags ...................................................................................... 29
A Flag Enablement Example ................................................................. 30
Adding a Flag to the Context Menu .............................................................. 31
Adding Flag Examples .................................................................................. 31
Running a Console Class .............................................................................. 35
Customizing Web Server Enablement .......................................................... 35
Customizing the View Bar Layout .............................................................. 36
View Bar Row Example ................................................................................. 37
Customizing the Automatic Download of Files At Log On ......................... 37
A Timestamp Tolerance Example ............................................................... 37
Overriding the Default Date/Time Format .................................................... 38
A Date and Time Override Example ............................................................ 38
Formatting Capabilities: The Time Format Syntax ....................................... 39
A Time and Date Format Example .............................................................. 41
Customizing Line Thickness ........................................................................ 41
Line Thickness Example .............................................................................. 41

Chapter 6. Creating a Demo
Capturing Live Views from your NetView Management Console System ............ 43
Using Basic Data Files ................................................................................. 45
Defining a Menu Item .................................................................................... 46
Defining a Set of Menus ............................................................................... 47
Defining a Resource Type. ............................................................................ 49
Defining Defaults ......................................................................................... 49
Creating a New Resource Type ..................................................................... 50
Integrating Captured Views into the Demo ................................................... 50
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 *Tivoli NetView for z/OS 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 z/OS:
- 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 web address 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 z/OS User ID and Password on the Topology Server

To use the optional *hostcmd* command to send commands from the topology server (using a command prompt on the topology server workstation) to the z/OS environment, specify a NetView for z/OS operator ID and password. There are three ways to define a NetView for z/OS 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 Windows: `%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 user id and password are accessed in this order.
Chapter 4. Customizing the NetView Management Console

Topology Server

This chapter describes customization of the NMC topology server to perform specific functions.

Customizing Topology Server Features

You can customize features of the NMC topology server, by modifying the server.properties file. One reason you might want to customize this file is to set the defaults for the audit log, as discussed in "Auditing Functions and 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
- Audit log attributes

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 Windows: %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.

Auditing Functions and the Server.Properties File

The server.properties file contains auditing function specifications that can be set to control the output and format of the auditing log.

Customizing the Functions to be Audited

You can determine which functions to audit in the server.properties file. The attribute for each of the functions that can be audited can have two possible values. If the attribute value is 1, the auditing is performed. If the attribute value is 0 (zero) then auditing is not performed. If an invalid value is assigned to the attribute, then the default value is used. Table 5 on page 20 describes the attributes and gives the default value for each.
Table 5. Auditing function attributes.

<table>
<thead>
<tr>
<th>Attribute in server.properties file</th>
<th>Description of function to be (or not to be) audited</th>
<th>Default value</th>
</tr>
</thead>
<tbody>
<tr>
<td>auditAggregationChange</td>
<td>aggregation priority and threshold change requests</td>
<td>0</td>
</tr>
<tr>
<td>auditCommand</td>
<td>listing commands to be executed</td>
<td>1</td>
</tr>
<tr>
<td>auditCommandResponse</td>
<td>listing responses to commands</td>
<td>0</td>
</tr>
<tr>
<td>auditConsoleSignonSignoff</td>
<td>signing on and off of the NMC topology console</td>
<td>1</td>
</tr>
<tr>
<td>auditCPESignonSignoff</td>
<td>signing on and off of the command profile editor (CPE) batch utility</td>
<td>1</td>
</tr>
<tr>
<td>auditFlagChange</td>
<td>flag change and list suspended resources requests</td>
<td>0</td>
</tr>
<tr>
<td>auditNetconvUpDown</td>
<td>executing the NETCONV command to start and stop communication between the host and server</td>
<td>1</td>
</tr>
<tr>
<td>auditSendMessage</td>
<td>sending messages</td>
<td>0</td>
</tr>
<tr>
<td>auditServerStartStop</td>
<td>starting and stopping of the NMC topology server</td>
<td>1</td>
</tr>
<tr>
<td>auditViewClose</td>
<td>closing views</td>
<td>0</td>
</tr>
<tr>
<td>auditViewCustomizationSaveRemove</td>
<td>saving, removing, and deleting customized views</td>
<td>0</td>
</tr>
<tr>
<td>auditViewOpenRequest</td>
<td>requesting views from the NetView host</td>
<td>0</td>
</tr>
<tr>
<td>auditViewOpenResponse</td>
<td>opening views</td>
<td>0</td>
</tr>
</tbody>
</table>

Customizing the Audit Log for Viewing

The ihsaudit.xml file is the log file which contains the data specified by the attributes in the server.properties file as described in the section "Customizing the Functions to be Audited" on page 19.

For example, an ihsaudit.xml file may contain audit entries indicating the following: start and initialization of an NMC topology server, netconv communication from the NMC topology server to a host, and an NMC topology console signing on to the NMC topology server. The following example contains such audit entries:

```xml
<?xml version="1.0" encoding="utf-8" ?>
<?xml-stylesheet href="../sample/ihsaudit.css" type="text/css" ?>

<auditLog>
  <auditEntry>
    <date>2001-12-11 15:22:24</date>
    <nmcServer>
      <action>start</action>
    </nmcServer>
  </auditEntry>

  <auditEntry>
    <date>2001-12-11 15:23:09</date>
    <netconv>
      <action>up</action>
      <nvDomain>CNM01</nvDomain>
    </netconv>
  </auditEntry>
</auditLog>
```
Refer to "Appendix E. Auditing and the ihsaudit.xml File" on page 173 for information about elements in the ihsaudit.xml file.

The audit log can be viewed by a Web browser, or other application, which supports XML. To enable a Web browser or other XML application to display the contents of the audit log properly, header lines must be inserted in the ihsaudit.xml log file. These lines are set in the server.properties file and determine the appropriate style sheet to be used for the browser. The header lines are inserted only when the Topology Server creates a new ihsaudit.xml file, not every time an audit entry is written to the file.

Note: Ensure that your Web browser or XML application supports the format of the style sheet that you choose.

The audit log file is located in one of the following directories:
- For Windows: %BINDIR%\TDS\server\log
- For UNIX: $BINDIR/TDS/server/log

Table 6 describes the XML attributes in the server.properties file.

<table>
<thead>
<tr>
<th>Attribute in server.properties file</th>
<th>Description of function</th>
<th>Default value</th>
</tr>
</thead>
<tbody>
<tr>
<td>auditLogHeaderLine.1</td>
<td>Sets the XML version</td>
<td>1.0</td>
</tr>
<tr>
<td>auditLogHeaderLine.2</td>
<td>Sets the style sheet to be used to enable the appropriate Web browser or XML application</td>
<td>ihsaudit.css</td>
</tr>
</tbody>
</table>

Example

```xml
auditLogHeaderLine.1 =&gt;<?xml version="1.0" encoding="utf-8" ?>
auditLogHeaderLine.2 =&gt;<?xml -stylesheet href="../sample/ihsaudit.css" type="text/css" ?>
```
Customizing the Size of the Audit Log

When the audit log file reaches the size specified with the auditLogFileMaxSize attribute in the server.properties file, it is renamed ihsaudit.bak. A new ihsaudit.xml file is then created. It is possible for the audit log to be slightly greater in size than specified by the auditLogFileMaxSize attribute. Audit entries are written to the log until the file size is equal to or greater than the value of auditLogFileMaxSize. For example, if the actual size of the file is 999 999 and the auditLogFileMaxSize is set to 1 000 000, then one more audit log entry is written to the file. Table 7 describes the auditLogFileMaxSize attribute in the server.properties file.

Table 7. Audit log size attribute set in the server.properties file.

<table>
<thead>
<tr>
<th>Attribute in server.properties file</th>
<th>Description</th>
<th>Default value</th>
<th>Valid values</th>
</tr>
</thead>
<tbody>
<tr>
<td>auditLogFileMaxSize</td>
<td>Determines the size of the audit log file, in megabytes</td>
<td>20</td>
<td>1–1000 (1 megabyte to 1 gigabyte)</td>
</tr>
</tbody>
</table>

If the value for the auditLogFileMaxSize attribute is set to 20, then the maximum file size is actually 20 000 000 bytes.

Notes:
1. If an incorrect value is entered for the auditLogFileMaxSize attribute, the default value is used.
2. If there was a previous ihsaudit.bak file, that file is deleted and replaced with the most current ihsaudit.xml backed up and renamed ihsaudit.bak.

Customizing the Length of Command Responses within the Audit Log

Audit log entries for command responses are written to the audit log if the auditCommandResponse value specified in the server.properties file is set to 1. The auditCommandResponse attribute is described in "Customizing the Functions to be Audited" on page 19. The maximum line length of these command responses can be specified in the server.properties file with the auditCommandResponseMaxSize attribute. When command responses exceed the length specified by this attribute, they are truncated in the audit log. Table 8 describes the auditCommandResponseMaxSize attribute in the server.properties file.

Table 8. Command response length attribute set in the server.properties file.

<table>
<thead>
<tr>
<th>Attribute in server.properties file</th>
<th>Description</th>
<th>Default value</th>
<th>Valid values</th>
</tr>
</thead>
<tbody>
<tr>
<td>auditCommandResponseMaxSize</td>
<td>Determines the maximum line length of command responses, in bytes, written to the audit log</td>
<td>1000</td>
<td>0–10 000</td>
</tr>
</tbody>
</table>

Notes:
1. The 0 (zero) value indicates that the command response is not truncated.
2. If an incorrect value is entered, the default value 1 000, is used.
3. If the command response contains an ampersand, (&), then the truncated string could be a few characters shorter or longer than the auditCommandResponseMaxSize indicates. If an ampersand (&) has been
translated to a string of characters that can be browsed (&amp;), for example, then the complete string is included in the truncated command response, regardless of the length of the string.

**Customizing Special Characters to be Browsed**

Without using the auditTranslateToBrowsableChars attribute in the `server.properties` file, the following set of characters can cause an error so that the ihsaudit.xml file cannot be opened in a browser:

- & (ampersand)
- < (less than)
- > (greater than)

The `auditTranslateToBrowsableChars` changes these characters to a string of characters that can be browsed within the following XML tags:

- `<cmd>`
- `<cmdResp>`
- `<locateName>`
- `<menuText>`
- `<msg>`
- `<note>`
- `<viewName>`

The `auditTranslateToBrowsableChars` attribute in the `server.properties` file translates these characters so that the ihsaudit.xml file can be viewed by a browser. The `auditTranslateToBrowsableChars` attribute in the `server.properties` file is described in [Table 9](#).

**Table 9. Characters to be browsed attribute set in the server.properties file.**

<table>
<thead>
<tr>
<th>Attribute in server.properties file</th>
<th>Description</th>
<th>Default value</th>
<th>Valid values</th>
</tr>
</thead>
<tbody>
<tr>
<td>auditTranslateToBrowsableChars</td>
<td>When set to 1, this attribute translates the characters as follows: • &amp; to &amp;  • &lt; to &lt;  • &gt; to &gt;  When set to 0, these characters are not translated.</td>
<td>1</td>
<td>0 or 1</td>
</tr>
</tbody>
</table>

**Note:** If a value other than 0 or 1 is entered for this attribute, the default (1) is used.

**Customizing the Replacement Character for Carriage Returns and Line Feeds**

Command responses often contain carriage return and line feed characters. Some browsers are unable to display an audit entry that contains these characters. However, you can choose to replace the carriage return and line feed characters with a character or multiple characters of your choice. To insert characters for the carriage return and line feed characters, use the `auditCarriageReturnReplacement`
attribute and specify the characters to be used for delineation with the

\texttt{auditCarriageReturnReplacementValue} attribute in the \texttt{server.properties} file as
described in Table 10.

\textbf{Table 10. Line return delineation attribute set in the server.properties file.}

<table>
<thead>
<tr>
<th>Attribute in server.properties file</th>
<th>Description</th>
<th>Default value</th>
<th>Valid values</th>
</tr>
</thead>
<tbody>
<tr>
<td>\texttt{auditCarriageReturnReplacement}</td>
<td>Determines if the carriage returns and line feeds are to be replaced with a character or characters</td>
<td>1</td>
<td>0 and 1</td>
</tr>
<tr>
<td>\texttt{auditCarriageReturnReplacementValue}</td>
<td>If the value of \texttt{auditCarriageReturnReplacement} is 1, determines the character(s) with which to replace the carriage returns and line feeds.</td>
<td>null (this field is empty)</td>
<td>0 to 25 bytes</td>
</tr>
</tbody>
</table>

When the \texttt{auditCarriageReturnReplacement} value is set to 1, the \texttt{auditCarriageReturnReplacementValue} is examined and the value assigned to the \texttt{auditCarriageReturnReplacementValue} attribute is used as the delimiter between lines. If the \texttt{auditCarriageReturnReplacementValue} attribute is not assigned a value (but the \texttt{auditCarriageReturnReplacement} attribute value is set to 1) then the carriage return and line feed characters are replaced with a single blank character.

When the value of \texttt{auditCarriageReturnReplacement} is set to 0, then no replacements occur and the \texttt{auditCarriageReturnReplacementValue} is ignored.

\textbf{Notes:}

1. The value assigned to the \texttt{auditCarriageReturnReplacementValue} field is not interpreted but is used as it appears. For example, if the \texttt{auditCarriageReturnReplacementValue} value is set to \texttt{/n}, then the two characters \texttt{/n} replace the carriage return and line feed characters; this does not cause a line return.

2. If the value assigned to the \texttt{auditCarriageReturnReplacementValue} attribute is longer than 25 characters, the string is truncated to the first 25 characters.
Chapter 5. Customizing the NetView Management Console

Topology Console

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

Customizing Topology Console Features

You can customize features of the NMC 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 Windows: %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 Windows: `%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 Windows: `%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:
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 `ihssce.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, ensuring that you set the port number ranges so that each console on the same machine can have a unique port number. 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 Web address in the Web browser. You do not need to add prefixes (http, www, and so on), although you might need to fully qualify the host name. If the port number for the Web server is not 80, follow the host name with a colon (:) and the port number in the browser as well. The port number for the Web server is displayed on the Web Server page of the Console Properties notebook. If the port number is 80, the Web address should appear as follows:

Clientname

If the port number is not 80, the Web address should appear as follows:

Clientname:Port number

You will not see a view in the Web browser until you add that view to the set of available views on the console. The view is displayed in the Web browser after the next Web browser refresh interval and the following information is displayed 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 is displayed 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 Web address is substituted for the $URL automatically. If you do not specify a $URL, the Help screen Web address 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:
      ```
      * User1
      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 define 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.

```java
// "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.forReal Applicable for a real resource? (optional, true)
// fx.forAgg Applicable for an aggregate resource? (optional, false)
// fx.relatedTo Mask of "related" flags (optional, none).
```

For more information on self-documenting data classes for the NMC topology console, see "Running a Console Class" on page 35.

6. Save your changes.
7. To define the wording for this user flag, open the defaultscheme.text.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 topology console.
2. Open a view.
3. Select a resource and right-click to display its context menu.
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 11 on page 31 shows the changes that were made to enable the User 1 flag and name it Retired.
Table 11. Changing the User 1 flag to retired.

<table>
<thead>
<tr>
<th>File</th>
<th>Before</th>
<th>After</th>
</tr>
</thead>
<tbody>
<tr>
<td>defaultscheme.properties</td>
<td>* User 1</td>
<td>* User 1</td>
</tr>
<tr>
<td></td>
<td>f25.value = 0x000000080</td>
<td>f25.value = 0x000000080</td>
</tr>
<tr>
<td></td>
<td>f25.weight = 2000</td>
<td>f25.weight = 2000</td>
</tr>
<tr>
<td></td>
<td>f25.defDefine = false</td>
<td>f25.defDefine = true</td>
</tr>
<tr>
<td>defaultschemetext.properties</td>
<td>f.00000080.label = User 1</td>
<td>f.00000080.label = Retired</td>
</tr>
<tr>
<td></td>
<td>f.00000080.abbrev = User1</td>
<td>f.00000080.abbrev = Retd</td>
</tr>
</tbody>
</table>

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 12.

Table 12. Description of NetView management console properties files

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

The disk location of the scheme properties files is determined by the NetView management console operational mode (as shown in Table 13 on page 32).
Table 13. Where to edit scheme files, based on mode.

<table>
<thead>
<tr>
<th>NetView management console Operational Mode</th>
<th>Location of Scheme Files</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signed on to a server</td>
<td><code>&lt;installRoot&gt;\bin\&lt;interp&gt;TDS\Server\db\current\settings</code></td>
</tr>
<tr>
<td>Demo</td>
<td><code>&lt;installRoot&gt;\bin\generic_unix\TDS\client\settings</code></td>
</tr>
</tbody>
</table>

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 `Userstatus 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 `menuX` 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.define 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 topology console, see "Running a Console Class" on page 33.

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 menuX 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 topology 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 z/OS Data Model Reference manual for details.

**Adding Flag Examples**
The example in Table 14 on page 34 shows the changes made to enable context menu items for Markedflag in the defaultscheme.properties file.
Table 14. The Markedflag item in the defaultschemetext.properties file.

<table>
<thead>
<tr>
<th>Before</th>
<th>After</th>
</tr>
</thead>
<tbody>
<tr>
<td>*********************************</td>
<td>*********************************</td>
</tr>
<tr>
<td>* User status menu items</td>
<td>* User status menu items</td>
</tr>
<tr>
<td>*********************************</td>
<td>*********************************</td>
</tr>
<tr>
<td>menu1.setRelTo = false</td>
<td>menu1.setRelTo = false</td>
</tr>
<tr>
<td>menu2.tag = SuspAut</td>
<td>menu2.tag = SuspAut</td>
</tr>
<tr>
<td>menu2.value = 0x40000000</td>
<td>menu2.value = 0x40000000</td>
</tr>
<tr>
<td>menu2.setTo = true</td>
<td>menu2.setTo = true</td>
</tr>
<tr>
<td>menu2.setRel = true</td>
<td>menu2.setRel = true</td>
</tr>
<tr>
<td>menu2.setRelTo = true</td>
<td>menu2.setRelTo = true</td>
</tr>
<tr>
<td>menu3.tag = ClearSusp</td>
<td>menu3.tag = ClearSusp</td>
</tr>
<tr>
<td>menu3.value = 0x40000000</td>
<td>menu3.value = 0x40000000</td>
</tr>
<tr>
<td>menu3.setTo = false</td>
<td>menu3.setTo = false</td>
</tr>
<tr>
<td>menu3.setRel = true</td>
<td>menu3.setRel = true</td>
</tr>
<tr>
<td>menu3.setRelTo = false</td>
<td>menu3.setRelTo = false</td>
</tr>
<tr>
<td>menu4.tag = ClearChildSusp</td>
<td>menu4.tag = ClearChildSusp</td>
</tr>
<tr>
<td>menu4.value = 0x00800000</td>
<td>menu4.value = 0x00800000</td>
</tr>
<tr>
<td>menu4.setTo = false</td>
<td>menu4.setTo = false</td>
</tr>
</tbody>
</table>

The example in Table 15 on page 35 shows the changes that were made to enable context menu items for the Markedflag in the defaultschemetext.properties file.
### Table 15. The Markedflag item in the defaultschemetext.properties file.

<table>
<thead>
<tr>
<th>Before</th>
<th>After</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>*******************************</code></td>
<td><code>*******************************</code></td>
</tr>
<tr>
<td>* User status menu items text</td>
<td>* User status menu items text</td>
</tr>
<tr>
<td><code>------------------------------------------------------------------------</code></td>
<td><code>------------------------------------------------------------------------</code></td>
</tr>
<tr>
<td>menu1.text = Suspend, Manually Clear</td>
<td>menu1.text = Suspend, Manually Clear</td>
</tr>
<tr>
<td>menu1.help = ihs_mi_clermansusp_$$$$$$.html</td>
<td>menu1.help = ihs_mi_clermansusp_$$$$$$.html</td>
</tr>
<tr>
<td>menu2.text = Suspend, Automatically Clear</td>
<td>menu2.text = Suspend, Automatically Clear</td>
</tr>
<tr>
<td>menu2.help = ihs_mi_clerautosusp_$$$$$$.html</td>
<td>menu2.help = ihs_mi_clerautosusp_$$$$$$.html</td>
</tr>
<tr>
<td>menu3.text = Clear Suspended</td>
<td>menu3.text = Clear Suspended</td>
</tr>
<tr>
<td>menu3.help = ihs_mi_clersuspended_$$$$$$.html</td>
<td>menu3.help = ihs_mi_clersuspended_$$$$$$.html</td>
</tr>
<tr>
<td>menu4.text = Clear Child Suspended</td>
<td>menu4.text = Clear Child Suspended</td>
</tr>
<tr>
<td>menu4.help = ihs_mi_clerchldsusp_$$$$$$.html</td>
<td>menu4.help = ihs_mi_clerchldsusp_$$$$$$.html</td>
</tr>
<tr>
<td>menu5.text = Set Marked</td>
<td>menu5.text = Set Marked</td>
</tr>
<tr>
<td>menu6.text = Clear Marked</td>
<td>menu6.text = Clear Marked</td>
</tr>
</tbody>
</table>

### Running a Console Class

Many of the NMC topology 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 topology 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 16 shows the values for enabling or disabling the Web browser function; these values are set in the defaultscheme.properties file.

#### Table 16. Enabling and disabling the web server function in the defaultscheme.properties file.

<table>
<thead>
<tr>
<th>defaultscheme.properties Attribute</th>
<th>Default Value</th>
<th>Other valid value</th>
</tr>
</thead>
<tbody>
<tr>
<td>webServerPage.enable</td>
<td>true</td>
<td>false</td>
</tr>
</tbody>
</table>

To customize the Web Server enablement, the NetView management console operational scheme must be updated, as shown in Table 16. 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.
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 topology 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 12 on page 31.

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 17 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>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Default Value</th>
<th>Minimum Value</th>
<th>Maximum Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>viewbar_max_rows = &lt;int&gt;</td>
<td>Maximum number of rows in the view bar grid. New columns will be added to display additional views.</td>
<td>0</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>viewbar_max_columns = &lt;int&gt;</td>
<td>Maximum number of columns in the view bar grid. New rows will be added to display additional views.</td>
<td>0</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>viewbar_min_view_size = &lt;int&gt;</td>
<td>Minimum size to make a view when sizing the views to fit on the view bar.</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
</tbody>
</table>

Table 17. Changing View Bar attributes and values in the defaultscheme.properties file.
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 13 on page 33 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 18 shows the changes made to the maximum number of rows in the View Bar grid from 0 to 5.

<table>
<thead>
<tr>
<th>Before</th>
<th>After</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>viewbar_max_rows = 0</code></td>
<td><code>viewbar_max_rows = 5</code></td>
</tr>
</tbody>
</table>

For more information on the NetView management console operational scheme, see Table 12 on page 33.

**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 F. Automatic File Download at Console Log On” on page 197 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 19 shows the attribute and the range of valid values for this attribute.

<table>
<thead>
<tr>
<th>defaultscheme.properties attribute</th>
<th>Default Value</th>
<th>Minimum Value</th>
<th>Maximum Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>file_download_timestamp_tolerance = int</code></td>
<td>720 (12 hours)</td>
<td>0</td>
<td>43,200 (30 days)</td>
</tr>
</tbody>
</table>

**A Timestamp Tolerance Example**

The example in Table 20 on page 38 shows two scenarios in which the timestamp tolerance can be used and the associated values.
Table 20. Examples of timestamp tolerance settings in the defaultscheme.properties file

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

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™ 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 21 shows the attributes (in the defaultschemetext.properties file) used to make this change.

Table 21. Changing the date/time format in the defaultschemetext.properties file.

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>override.date = &lt;format&gt;</td>
<td>format used when just formatting a date</td>
</tr>
<tr>
<td>override.time = &lt;format&gt;</td>
<td>format used when just formatting a time</td>
</tr>
<tr>
<td>override.datetime = &lt;format&gt;</td>
<td>format used when formatting a composite date</td>
</tr>
<tr>
<td></td>
<td>and time</td>
</tr>
</tbody>
</table>

The example in Table 22 shows how a time stamp of 6:45:07 PM on March 30, 2001 displays for various format specifications.

Table 22. An example of time stamp format specifications.

<table>
<thead>
<tr>
<th>Format Specification</th>
<th>Resulting Display Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>yyyy.MM.dd</td>
<td>2001.03.30</td>
</tr>
<tr>
<td>yyyy.MM.dd</td>
<td>2001.Mar.30</td>
</tr>
<tr>
<td>yyyy.MMMM.dd</td>
<td>2001.March.30</td>
</tr>
<tr>
<td>hh:mm:ss a</td>
<td>06:45:07 PM</td>
</tr>
<tr>
<td>hh:mm:ss a z</td>
<td>06:45:07PM EST</td>
</tr>
<tr>
<td>HH:mm:ss z</td>
<td>18:45:07 EST</td>
</tr>
</tbody>
</table>
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 13 on page 32 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 topology 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 23 shows the changes made to override all of the date/time display formats. These changes were made in the defaultschemetext.properties file.

<table>
<thead>
<tr>
<th>Before</th>
<th>After</th>
</tr>
</thead>
<tbody>
<tr>
<td>****************************</td>
<td>****************************</td>
</tr>
<tr>
<td>* Date/Time Overrides Formats</td>
<td>* Date/Time Overrides Formats</td>
</tr>
<tr>
<td>****************************</td>
<td>****************************</td>
</tr>
<tr>
<td>* override.date =yyyy.MMM.dd</td>
<td>override.date =yyyy.MMM.dd</td>
</tr>
<tr>
<td>* override.time =HH:mm:ss z</td>
<td>override.time =HH:mm:ss z</td>
</tr>
<tr>
<td>* override.datetime=yyyyy.MMM.dd @</td>
<td>override.datetime=yyyyy.MMM.dd @</td>
</tr>
<tr>
<td>HH:mm:ss z</td>
<td>HH:mm:ss z</td>
</tr>
</tbody>
</table>

For more information on the NetView management console operational scheme, see Table 12 on page 31.

Formatting Capabilities: The Time Format Syntax
The example in Table 24 on page 40 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 24 on page 40.
Table 24. The time format syntax: formatting capabilities.

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

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 25 use a US Location.

**Table 25. Examples of time and date formatting.**

<table>
<thead>
<tr>
<th>Format Pattern</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;yyyy.MM.dd G 'at' hh:mm:ss z&quot;</td>
<td>2001.04.01 AD at 15:08:56 PDT</td>
</tr>
<tr>
<td>&quot;EEE, MMM d, ''yy&quot;</td>
<td>Sun, April 01, '01</td>
</tr>
<tr>
<td>&quot;h:mm a&quot;</td>
<td>12:08 PM</td>
</tr>
<tr>
<td>&quot;hh 'o'clock' a, zzzz&quot;</td>
<td>12 o'clock PM, Pacific Daylight Time</td>
</tr>
<tr>
<td>&quot;K:mm a, z&quot;</td>
<td>0:00 PM, PST</td>
</tr>
<tr>
<td>&quot;yyyyy.MMMMM.dd GGG hh:mm aaa&quot;</td>
<td>2001.April.01 AD 12:08 PM</td>
</tr>
</tbody>
</table>

### Customizing Line Thickness

You can adjust the thickness of lines in a topology view. Table 26 shows the values for line thickness; these values are set in the defaultscheme.properties file.

**Table 26. Changing line thickness (in pixels).**

<table>
<thead>
<tr>
<th>defaultscheme.properties Attribute</th>
<th>Default Value (in pixels)</th>
<th>Minimum Value (in pixels)</th>
<th>Maximum Value (in pixels)</th>
</tr>
</thead>
<tbody>
<tr>
<td>line_thickness</td>
<td>2</td>
<td>1</td>
<td>4</td>
</tr>
</tbody>
</table>

To customize the line thickness, the NetView management console operational scheme must be updated, as shown in Table 26. This can be done from the console or the server workstation, depending on the mode in which you are running. See Table 13 on page 32 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 topology console.
2. Open a view.
3. Verify the line thickness while in the topology view.

### Line Thickness Example

The example in Table 27 on page 42 shows the changes made to the line thickness from two pixels to one pixel.
Table 27. Changing the line thickness in the defaultscheme.properties file.

<table>
<thead>
<tr>
<th>Before</th>
<th>After</th>
</tr>
</thead>
<tbody>
<tr>
<td>***************************************</td>
<td>***************************************</td>
</tr>
<tr>
<td>* Thickness of connection lines in pixels</td>
<td>* Thickness of connection lines in pixels</td>
</tr>
<tr>
<td>***************************************</td>
<td>***************************************</td>
</tr>
<tr>
<td>line_thickness = 2</td>
<td>line_thickness = 1</td>
</tr>
</tbody>
</table>

For more information on the NetView management console operational scheme, see Table 12 on page 31.
Chapter 6. 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.

The demo capability is installed as part of the NetView management console Productivity Kit (on the Windows platform, only). Be sure to perform a custom installation of the NMC topology console and select the NetView management console Productivity Kit. To access further information about the demo function, in the Windows environment, double-click the NMC Productivity Kit icon on the desktop.

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.

2. "Integrating Captured Views into the Demo" on page 50 by:
   a. "Updating the Business Tree" on page 50.

Once these steps are completed, the files shown in Table 28 will comprise the new demo.

Table 28. Demo files.

<table>
<thead>
<tr>
<th>Demo</th>
<th>Directory</th>
<th>Files</th>
</tr>
</thead>
<tbody>
<tr>
<td>View files (basic data files)</td>
<td>&lt;installRoot&gt;\bin\generic_unix\TDS\client\views</td>
<td>*.md</td>
</tr>
<tr>
<td>Basic Data definitions (See &quot;Using Basic Data Files&quot; on page 43 to define additional resource types).</td>
<td>&lt;installRoot&gt;\bin\generic_unix\TDS\client\settings</td>
<td>basicdata.nmc</td>
</tr>
</tbody>
</table>

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 52.

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 topology 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 topology 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 54.
   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 51.
   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 52.

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 topology 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 <output_dir>\filename
```

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

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

### Defining a Menu Item

Table 29 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 30.

**Table 29. Description of menu item attributes.**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>mi.N.&lt;attr&gt;</td>
<td>Java applications launched using com.tivoli.ihs.client.action.IhsJavaAppAction. <strong>Note:</strong> Use this type of attribute for testing Java applications.</td>
</tr>
<tr>
<td>sc.N.&lt;attr&gt;</td>
<td>Predefined scenarios.</td>
</tr>
<tr>
<td>nop.N.&lt;attr&gt;</td>
<td>NOP menus which appear in the context menu but do not do anything when selected.</td>
</tr>
</tbody>
</table>

**Table 30. Attributes that can be specified for a menu item.**

<table>
<thead>
<tr>
<th>&lt;attr&gt;</th>
<th>Description</th>
<th>mi.N.&lt;attr&gt;</th>
<th>Sc.N.&lt;attr&gt;</th>
<th>nop.N.&lt;attr&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>tag</td>
<td>Tag name that is used to reference this menu item</td>
<td>Required</td>
<td>Required</td>
<td>Required</td>
</tr>
<tr>
<td>label</td>
<td>Menu item text</td>
<td>Required</td>
<td>Required</td>
<td>Required</td>
</tr>
<tr>
<td>class</td>
<td>Package-qualified Java class that provides the Java implementation of this menu item</td>
<td>Optional</td>
<td>Ignored</td>
<td>Ignored</td>
</tr>
<tr>
<td></td>
<td>Def: IhsJavaAppAction¹</td>
<td>Def: IhsScenario³</td>
<td>Def: IhsCPEAction¹</td>
<td></td>
</tr>
<tr>
<td>data</td>
<td>Class specific data</td>
<td>Optional</td>
<td>Required</td>
<td>Ignored</td>
</tr>
<tr>
<td></td>
<td>Def: blank</td>
<td>scenario file name and arguments</td>
<td></td>
<td></td>
</tr>
<tr>
<td>html</td>
<td>Menu Help HTML panel name for the menu item</td>
<td>Optional</td>
<td>Optional</td>
<td>Optional</td>
</tr>
<tr>
<td></td>
<td>Def: ihs_nohelp_XXX.html</td>
<td>Def: ihs_nohelp_XXX.html</td>
<td>Def: ihs_nohelp_XXX.html</td>
<td></td>
</tr>
</tbody>
</table>
Table 30. Attributes that can be specified for a menu item. (continued)

<table>
<thead>
<tr>
<th>&lt;attr&gt;</th>
<th>Description</th>
<th>mi.N.&lt;attr&gt;</th>
<th>Sc.N.&lt;attr&gt;</th>
<th>nop.N.&lt;attr&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>max</td>
<td>Maximum number of selected resource(s) supported by a particular resource type</td>
<td>Optional</td>
<td>Optional</td>
<td>Optional</td>
</tr>
<tr>
<td></td>
<td>Def: 1</td>
<td>Def: 1</td>
<td>Def: 1</td>
<td></td>
</tr>
<tr>
<td>debug</td>
<td>Determines if this is a debug menu item; enabled when the -debug command line argument is used</td>
<td>Optional</td>
<td>Optional</td>
<td>Optional</td>
</tr>
<tr>
<td></td>
<td>Def: false</td>
<td>Def: false</td>
<td>Def: false</td>
<td></td>
</tr>
<tr>
<td>vfy</td>
<td>Determines if a verification prompt should appear before item is launched ²</td>
<td>Optional</td>
<td>Optional</td>
<td>Optional</td>
</tr>
<tr>
<td></td>
<td>Def: false</td>
<td>Def: false</td>
<td>Def: false</td>
<td></td>
</tr>
</tbody>
</table>

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 31 can be specified for a set.

Table 31. Attributes that can be specified for a set.

<table>
<thead>
<tr>
<th>&lt;attr&gt;</th>
<th>Description</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>tag</td>
<td>Name used to reference this set</td>
<td>Required</td>
</tr>
</tbody>
</table>
### Table 31. Attributes that can be specified for a set. (continued)

<table>
<thead>
<tr>
<th><strong>&lt;attr&gt;</strong></th>
<th><strong>Description</strong></th>
<th><strong>Default</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>label</td>
<td>Name of cascaded menu item</td>
<td>Optional</td>
</tr>
<tr>
<td></td>
<td>- If omitted, the set is used as a container; items are copied into the current menu.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- If specified, a cascaded menu item is created with the menu items for this set.</td>
<td></td>
</tr>
<tr>
<td>debug</td>
<td>Determines if this is a debug set; enabled when the -debug command line argument is used</td>
<td>Optional Def: false</td>
</tr>
</tbody>
</table>

Each item of a set references one of the items in **Table 32**.

### Table 32. Set reference items.

<table>
<thead>
<tr>
<th><strong>Tag</strong></th>
<th><strong>Reference</strong></th>
<th><strong>Description</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>- (dash)</td>
<td>Menu separator</td>
<td></td>
</tr>
<tr>
<td>tag name</td>
<td>Reference to a:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• menu item</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• set</td>
<td></td>
</tr>
</tbody>
</table>

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

```plaintext
###################################################
# Context menu items for an AGGREGATE resource
# Constant menu items for a specific AGGREGATE resource
set.3.tag =@nmcAgg
set.3.1 =@baseAgg
set.3.2 =-
set.3.3 =nmcFailing
# "Configuration" cascaded menu
# Constant menu items for a specific AGGREGATE resource
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
# Constant menu items for a specific AGGREGATE resource
set.5.tag =@exJavaApps
set.5.debug =true
set.5.1 =-
set.5.2 =exGUI
set.5.3 =exCmdRsp
set.5.4 =exIPL
```

---

48   NetView Management Console User’s Guide
Defining a Resource Type

A resource type is defined using syntax as shown in Table 33.

Table 33. Attributes that can be specified for a resource type.

<table>
<thead>
<tr>
<th>&lt;attr&gt;</th>
<th>Description</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>rtX_name</td>
<td>Name; specified in node/link &quot;resource type&quot; field</td>
<td>rtX_desc</td>
</tr>
<tr>
<td>rtX_desc</td>
<td>Description text</td>
<td>Optional resource type name</td>
</tr>
<tr>
<td>rtX_image</td>
<td>Image file name</td>
<td>Optional node.gif</td>
</tr>
<tr>
<td>rtX_geometric</td>
<td>Geometric shape name</td>
<td>Optional com.tivoli.ihs.reuse.gui.IhsRectangle</td>
</tr>
<tr>
<td>rtX_help</td>
<td>Help URL</td>
<td>Optional none</td>
</tr>
<tr>
<td>rtX_flags</td>
<td>Flags</td>
<td>Optional 1</td>
</tr>
</tbody>
</table>

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=instSsmtpGW

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"
###########################################################
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 basic data 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 54.
   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 34. For more detailed information about the view
   model, see "Defining View Information" on page 56.

   Table 34. Replacing the view model text example (before and after).

<table>
<thead>
<tr>
<th>Before</th>
<th>After</th>
</tr>
</thead>
<tbody>
<tr>
<td>view_id=xxxx</td>
<td>view_id=availAvail</td>
</tr>
<tr>
<td>view_prev_id=-1</td>
<td>view_width=500</td>
</tr>
<tr>
<td>view_useimage=-1</td>
<td>view_height=900</td>
</tr>
<tr>
<td>view_openview=0</td>
<td>view_customizable=true</td>
</tr>
</tbody>
</table>

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 35.

   Table 35. Updating the resource type example (before and after).

<table>
<thead>
<tr>
<th>Before</th>
<th>After</th>
</tr>
</thead>
<tbody>
<tr>
<td>anx_t=536871171</td>
<td>anx_t=RT536871171</td>
</tr>
</tbody>
</table>

6. Verify that the special business tree nodes shown in Table 36 on page 52 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 36 on page 52.

   This enables you to avoid error messages when NetView management console
   is started and have the correct icon displayed in the business tree.
Table 36. Verifying the resource type values for business tree nodes.

<table>
<thead>
<tr>
<th>Special Business Tree Node (found in the anX_l attribute)</th>
<th>Required Resource Type (defined in basicdata.nmc)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Business Systems</td>
<td>RT536871171</td>
</tr>
<tr>
<td>Systems Management Business System</td>
<td>RT536871228</td>
</tr>
<tr>
<td>Networking</td>
<td>RT536871168</td>
</tr>
<tr>
<td>Network Views</td>
<td>RT536871169</td>
</tr>
<tr>
<td>Exception Views</td>
<td>RT536871170</td>
</tr>
</tbody>
</table>

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 37 shows the naming conventions.

Table 37. Naming conventions for navigation views.

<table>
<thead>
<tr>
<th>Navigation Context Menu</th>
<th>Target File Naming Convention</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configuration &gt; Parents</td>
<td>&lt;resource ID&gt;.par.ctl.md</td>
</tr>
<tr>
<td>Configuration &gt; Children</td>
<td>&lt;resource ID&gt;.chd.ctl.md</td>
</tr>
<tr>
<td>Configuration &gt; Peers</td>
<td>&lt;resource ID&gt;.peer.ctl.md</td>
</tr>
<tr>
<td>Configuration &gt; Logical and Physical</td>
<td>&lt;resource ID&gt;.lp.ctl.md</td>
</tr>
<tr>
<td>Configuration &gt; Logical</td>
<td>&lt;resource ID&gt;.log.ctl.md</td>
</tr>
<tr>
<td>Configuration &gt; Physical</td>
<td>&lt;resource ID&gt;.phy.ctl.md</td>
</tr>
<tr>
<td>Configuration &gt; Backbone</td>
<td>&lt;resource ID&gt;.bak.ctl.md</td>
</tr>
<tr>
<td>Locate Failing Resources</td>
<td>&lt;resource ID&gt;.fp.ctl.md</td>
</tr>
<tr>
<td>Locate Resource</td>
<td>&lt;locate name&gt;.locate.md</td>
</tr>
<tr>
<td>More Detail</td>
<td>&lt;resource ID&gt;.ctl.md</td>
</tr>
</tbody>
</table>
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 a1Xt attribute of a link resource.
- 3 is the resource ID specified in the a1X_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 45. For more information on creating new resource types, see "Creating a New Resource Type" on page 50.
4. Using the information in Table 38, create a new resource type at the end of the currently defined set.

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

The following example shows how an APPN® end node would be defined:
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 39.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>anX_i</td>
<td>Display ID (unique per view).</td>
</tr>
<tr>
<td>anX_r</td>
<td>Server ID (unique per server); double clicking the resource drills to &lt;serverID&gt;.md.</td>
</tr>
<tr>
<td>anX_p</td>
<td>Parent ID (optional, none).</td>
</tr>
<tr>
<td>anX_t</td>
<td>Resource type (optional, defaulted).</td>
</tr>
<tr>
<td>anX_s</td>
<td>Status (optional, normal).</td>
</tr>
<tr>
<td>anX_u</td>
<td>Flags (optional, zero).</td>
</tr>
<tr>
<td>anX_a</td>
<td>Determines if this is an aggregate resource (optional, false).</td>
</tr>
<tr>
<td>anX_da</td>
<td>Suppress ‘+’ on aggregate resources. For Topology Display Subsystem view only, others not live if suppressed (optional, value of anX_a).</td>
</tr>
<tr>
<td>anX_l</td>
<td>Label (optional, blank).</td>
</tr>
<tr>
<td>anX lx</td>
<td>Label X coordinate (optional, auto).</td>
</tr>
<tr>
<td>anX_ly</td>
<td>Label Y coordinate (optional, auto).</td>
</tr>
</tbody>
</table>
Table 39. Attributes to define a node resource in a view. (continued)

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>anX_d1</td>
<td>Data 1: TDS=HB1, NMC=RODM other data (optional, blank).</td>
</tr>
<tr>
<td>anX_d2</td>
<td>Data 2: TDS=HB2, NMC=RODM customer data (optional, blank).</td>
</tr>
<tr>
<td>anX_d3</td>
<td>Data 3: TDS=HB3, NMC=not used (optional, blank).</td>
</tr>
<tr>
<td>anX_d4</td>
<td>Data 4: TDS=HB4, NMC=not used (optional, blank).</td>
</tr>
<tr>
<td>anX_per</td>
<td>On the business tree view, determines if a view is permanent (optional, false).</td>
</tr>
<tr>
<td>anX_cag</td>
<td>Specifies if customer aggregation is allowed (Not currently in use.) (optional, true).</td>
</tr>
<tr>
<td>anX_c1</td>
<td>Monitor Count 1 (optional, none).</td>
</tr>
<tr>
<td>anX_c2</td>
<td>Monitor Count 2 (optional, none).</td>
</tr>
<tr>
<td>anX_x</td>
<td>X coordinate.</td>
</tr>
<tr>
<td>anX_y</td>
<td>Y coordinate.</td>
</tr>
</tbody>
</table>

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 40.

Table 40. Attributes to define a link resource in a view.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>alX_i</td>
<td>Display ID (unique per view).</td>
</tr>
<tr>
<td>alX_r</td>
<td>Server ID (unique per server); double clicking the resource drills to&lt;serverID&gt;.md.</td>
</tr>
<tr>
<td>alX_p</td>
<td>Parent ID (optional, none).</td>
</tr>
<tr>
<td>alX_t</td>
<td>Resource type (optional, defaulted).</td>
</tr>
<tr>
<td>alX_s</td>
<td>Status (optional, normal).</td>
</tr>
<tr>
<td>alX_u</td>
<td>Flags (optional, zero).</td>
</tr>
<tr>
<td>alX_a</td>
<td>Determines if this is an aggregate resource (optional, false).</td>
</tr>
<tr>
<td>alX_da</td>
<td>Suppress ‘+’ on aggregate resources. For Topology Display Subsystem view only, others not &quot;real life&quot; if suppressed (optional, value of alX_a).</td>
</tr>
<tr>
<td>alX_l</td>
<td>Label (optional, blank).</td>
</tr>
<tr>
<td>alX lx</td>
<td>Label X coordinate (optional, auto).</td>
</tr>
<tr>
<td>alX_ly</td>
<td>Label Y coordinate (optional, auto).</td>
</tr>
<tr>
<td>alX_d1</td>
<td>Data 1: TDS=HB1, NMC=RODM other data (optional, blank).</td>
</tr>
<tr>
<td>alX_d2</td>
<td>Data 2: TDS=HB2, NMC=RODM customer data (optional, blank).</td>
</tr>
</tbody>
</table>
Table 40. Attributes to define a link resource in a view. (continued)

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>alX_d3</td>
<td>Data 3: TDS=HB3, NMC=not used (optional, blank).</td>
</tr>
<tr>
<td>alX_d4</td>
<td>Data 4: TDS=HB4, NMC=not used (optional, blank).</td>
</tr>
<tr>
<td>alX_per</td>
<td>On the business tree view, determines if a view is permanent (optional, false).</td>
</tr>
<tr>
<td>alX_cag</td>
<td>Specifies if customer aggregation is allowed (Not currently in use.) (optional, true).</td>
</tr>
<tr>
<td>alX_c1</td>
<td>Monitor Count 1 (optional, none).</td>
</tr>
<tr>
<td>alX_c2</td>
<td>Monitor Count 2 (optional, none).</td>
</tr>
<tr>
<td>alX_1</td>
<td>First end point able to be displayed.</td>
</tr>
<tr>
<td>alX_2</td>
<td>Second end point able to be displayed.</td>
</tr>
</tbody>
</table>
| 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 41 shows the attributes used to define the appearance of a view.

Table 41. Attributes to define appearance of a view.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>view_id</td>
<td>ID for the view</td>
</tr>
<tr>
<td>view_prev_id</td>
<td>Previous ID for the view</td>
</tr>
<tr>
<td>view_width</td>
<td>Width of view (optional, 500)</td>
</tr>
<tr>
<td>view_height</td>
<td>Height of view (optional, 300)</td>
</tr>
<tr>
<td>view_layer</td>
<td>Layer of view</td>
</tr>
<tr>
<td>view_descriptor</td>
<td>Descriptor of view (optional, ””)</td>
</tr>
<tr>
<td>view_fgcolor</td>
<td>Foreground text color of view; RGB value (optional, black)</td>
</tr>
<tr>
<td>view_ftcolor</td>
<td>Free text color of view; RGB value (optional, black)</td>
</tr>
<tr>
<td>view_useimage</td>
<td>Specifies whether to use image(1)/color(0)/notSet(-1) for view background (optional, -1)</td>
</tr>
<tr>
<td>viewbgcolor</td>
<td>If color, the background color RGB value (optional, gray)</td>
</tr>
<tr>
<td>view_bg</td>
<td>If image, specific image file name</td>
</tr>
<tr>
<td>view_bgx</td>
<td>If image, image X location within view (optional, -1)</td>
</tr>
<tr>
<td>view_bgy</td>
<td>If image, image Y location within view (optional, -1)</td>
</tr>
</tbody>
</table>
Table 41. Attributes to define appearance of a view. (continued)

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>view_bgwidth</td>
<td>If image, width (optional, -1)</td>
</tr>
<tr>
<td>view_bgheight</td>
<td>If image, height (optional, -1)</td>
</tr>
<tr>
<td>view_swidth</td>
<td>Width of resource symbol area (optional, 40)</td>
</tr>
<tr>
<td>view_sheight</td>
<td>Height of resource symbol area (optional, 40)</td>
</tr>
<tr>
<td>view_cust</td>
<td>Specifies if the view is customized (optional, false)</td>
</tr>
<tr>
<td>view_customizable</td>
<td>Specifies if the view can be customized (optional, true)</td>
</tr>
<tr>
<td>view_ffu</td>
<td>Specifies whether or not to force full update of view (optional, false)</td>
</tr>
<tr>
<td>view_orp</td>
<td>Specifies if override refresh property (optional, false)</td>
</tr>
<tr>
<td>view_activetab</td>
<td>View automatically opened for tab (optional, first tab)</td>
</tr>
<tr>
<td>view_openview</td>
<td>Open view option3 (topology or detail view)#</td>
</tr>
</tbody>
</table>

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

```java
view_id=89
view_prev_id=-1
viewbgcolor=-1
viewfgcolor=-16777216
viewftcolor=-16777216
```

### 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 43. It contains the following collection of object definitions:

1. A view model definition, as described in "Defining View Information" on page 56.
2. Zero or more node definitions, as described in "Defining a Node Resource in a View" on page 54.
3. Zero or more tack point 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 55.

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

```java
# 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
viewbgcolor=-1
viewfgcolor=-16777216
viewftcolor=-16777216
```
NetView Management Console User’s Guide
Chapter 7. 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 Windows platform, only). Be sure to perform a custom installation of the NMC topology console and select the NetView management console Productivity Kit. 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 to these examples are made here. The NetView management console Productivity Kit contains more detailed technical information (such as methods, fields, syntax and Java class hierarchy). To access this information, in the Windows environment, double-click the NMC Productivity Kit icon on the desktop.

Note: Though the NetView management console Productivity Kit can be installed only on the Windows platform, Java application and plug-in functions are available on any supported NMC topology console platform.

Supplied Support Files

The files shown in Table 42 provide additional support for the Java application and Java plug-in examples.

Table 42. Files that support the Java examples.

<table>
<thead>
<tr>
<th>File</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>examples\java\ExampleJavaApp.jar</td>
<td>This Java ARchive (JAR) file contains the compiled class files of the examples, so they are immediately usable.</td>
</tr>
<tr>
<td>examples\support\ExampleJavaApp.rsp</td>
<td>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.</td>
</tr>
<tr>
<td>examples\support\plugins.properties</td>
<td>This is a plug-in definition file that loads all of the example plug-ins.</td>
</tr>
<tr>
<td>examples\javadoc</td>
<td>This is the starting point for the console API documentation.</td>
</tr>
</tbody>
</table>

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 platform. See the Supplied Support Files section of the NetView management console Productivity Kit for more information.
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 42 on page 59 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.

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".
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
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.
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 the 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 10. Using the NetView Management Console Command Profile Editor](#) on page 93.

**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 43 are provided with NetView management console.

<table>
<thead>
<tr>
<th>File</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>examples\java\ExampleGUIJavaApp.java</td>
<td>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.</td>
</tr>
<tr>
<td>examples\java\ExampleCmdWithResponse.java</td>
<td>Illustrates sending a command to the Tivoli NetView for z/OS program and receiving all response lines for subsequent processing.</td>
</tr>
<tr>
<td>examples\java\ExampleGUItoIML.java</td>
<td>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 z/OS program. The results from the command are displayed in the console Log window.</td>
</tr>
</tbody>
</table>

**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 60.

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 61.

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 NMC topology 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.

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 66.
• 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 44 is displayed for each active plug-in. For more information, see the Java Plug-Ins section of the NetView management console Productivity Kit.

Table 44. Values for plug-ins as shown in the Environment Information window.

<table>
<thead>
<tr>
<th>Field Column</th>
<th>Value Column</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;plug-in class name&gt;:data</td>
<td>Initialization data passed to the setPlugInData() method.</td>
</tr>
<tr>
<td>&lt;plug-in class name&gt;:desc</td>
<td>Description as provided by the getPlugInDescription() method.</td>
</tr>
<tr>
<td>&lt;plug-in class name&gt;:version</td>
<td>Version as provided by the getPlugInVRM() method.</td>
</tr>
<tr>
<td>&lt;plug-in class name&gt;:debug</td>
<td>Debug information as provided by the toString() method.</td>
</tr>
</tbody>
</table>
  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 45 can be used to enhance base console operation.

<table>
<thead>
<tr>
<th>File</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>examples\java\ExampleLogPlugIn.java</td>
<td>Provides control over items (commands, responses, or messages) that are added to the console Log window. Also see &quot;Log Window Filter Plug-In&quot; on page 65.</td>
</tr>
<tr>
<td>examples\java\ExampleViewLabelPlugIn.java</td>
<td>Provides control over the label that is displayed for a resource. Also see &quot;View Label Formatter Plug-In&quot; on page 64.</td>
</tr>
</tbody>
</table>

**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 60.

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 61.

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 NMC topology 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 8. Configuring Property Files for Locally Launched Applications

The NetView management console provides the capability to launch local applications on the NMC topology console workstation, such as a Web browser, to view a specific Web address 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, on the NMC topology server workstation, 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.IhsCommandInvokercmdname 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:

```plaintext
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 = ( 
```
**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 69.

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 Windows: `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:

```
tappxx .. 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 157 for more information about the `tappxx` 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\<username>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 46 displays the information for the generic section of the properties file.

Table 46. Generic Section of Properties File

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>command.desc</td>
<td>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.</td>
</tr>
</tbody>
</table>

Table 47 displays the information for the browser section of the properties file.

Table 47. Browser Section of Properties File

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>browser.usebuiltin</td>
<td>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.</td>
</tr>
</tbody>
</table>

Table 48 on page 72 displays the information for the operating system section of the properties file.
<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>command.x.platform</td>
<td>Identifies the operating system for which this entry is valid. Valid operating systems include the following:</td>
</tr>
<tr>
<td></td>
<td>• Windows NT</td>
</tr>
<tr>
<td></td>
<td>• Windows 2000</td>
</tr>
<tr>
<td></td>
<td>• AIX</td>
</tr>
<tr>
<td></td>
<td>• Solaris</td>
</tr>
<tr>
<td></td>
<td>• HP-UX</td>
</tr>
<tr>
<td></td>
<td>• Linux</td>
</tr>
</tbody>
</table>

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>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>command.x.run</strong></td>
<td>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.</td>
</tr>
</tbody>
</table>

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.x.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 and Windows 2000 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.x.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.x.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.

---

Chapter 8. Configuring Property Files for Locally Launched Applications 73
Table 48. Operating System Section of Properties File (continued)

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
</table>
| 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 Web address to be displayed when the browser is started. The specific Web address is defined in the command response file. The substitution variables are gathered from three locations:

- 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 69 for the Web address 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. You can optionally specify a colon (:) and a token number to retrieve a single period-delimited token. For example, the MyName field of a NetView Resource Manager aggregate object in RODM is in the form: NRM.CNM01.AUTO1. If you want to extract just "AUTO1", you can specify: %RODM.Myname:3% This extracts the third period-delimited token.

The following example illustrates a properties file:

```properties
;-------------------------------------------------------------------------
; Your comments go here.
;-------------------------------------------------------------------------
browser.desc = Open Web browser and show URL
browser.usebuiltin = false

browser.1.platform = Windows *
browser.1.run = cmd /c start
browser.1.args = %url%

browser.2.platform = aix
browser.2.run = netscape
browser.2.args = %url%

browser.3.platform = *
browser.3.run = netscape
browser.3.args = %url%

;-------------------------------------------------------------------------
ping.1.platform = Windows *
ping.1.run = cmd /c start ping -t
ping.1.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 %d2cmdargs% 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 109 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 three specifications for telnet, numbered incrementally from 1 to 3. 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 9. Operating the NetView Management Console

Starting the Topology Server ........................................... 79
Starting the Topology Server from the Desktop Icon ................. 79
Manually Starting the Topology Server .................................. 79
Starting the Topology Server as a Windows Service .................. 80
Starting the Topology Server as a Daemon ............................ 80
Establishing Communication Between the NetView Host and the Topology Server ................................................. 80
Starting the Topology Console ............................................ 81
Selecting the Desktop Icon in Windows ................................. 81
Using a Line Command ...................................................... 81
Using the Topology Console Sign On Window ......................... 81
The Topology Console Window ............................................. 83
The View Area ..................................................................... 85
The Filter Bar ...................................................................... 86
NetView Management Console Online Help ............................. 86
NetView Management Console Features and Functions ............... 86
Issuing IP Commands .......................................................... 86
The NetView Resource Manager (NRM) .................................. 87
Using the RODM Collection Manager With NetView Management Console ......................................................... 88
NetView Management Console Topology Server Databases ........... 89
Writing Server Information to the Topology Server Databases ...... 89
When the Topology Server Databases Are Corrupted .................. 89
Creating and Restoring a Permanent Copy of the Topology Server Databases ......................................................... 90
Stopping the Topology Console ............................................. 90
Stopping the Topology Server .............................................. 90
With the Service Version on Windows .................................... 90
Using a Line Mode Command ............................................... 91

Chapter 10. Using the NetView Management Console Command Profile Editor

Understanding the Command Profile Editor ............................. 93
Resource Manager Objects .................................................. 93
Commands and Command Set Objects .................................... 93
Profile Objects ................................................................... 94
Operator Objects ................................................................ 94
Using the Command Profile Editor Batch Utility ....................... 95
Starting the Command Profile Batch Utility ............................. 95
Input and Output Files of the Response File ............................. 96
Manager Keywords ............................................................. 97
Command Keywords ........................................................... 97
Page Keywords in the Command Block ................................... 98
Command Set Keywords ....................................................... 101
Profile Keywords ............................................................... 103
Operator Keywords ............................................................ 103

Chapter 11. Using the Topology Server Command Exits

Command Profiles ............................................................... 105
Understanding Topology Server Command Exits ........................ 105
Using Topology Server Command Exits ................................... 106
IHSXGENCE Command Exit .................................................. 106
IHSXNATV Command Exit .................................................... 107
IHSXTHCE Command Exit .................................................... 107
IHSXJAM Command Exit ..................................................... 108
IHSXJAV Command Exit ..................................................... 108
Substitution Variables .......................................................... 108
%REMOTECONSOLE% ....................................................... 109
Writing Topology Server Command Exits ................................ 110
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Available Programming Languages</td>
<td>111</td>
</tr>
<tr>
<td>Command Exit Flow Scenario</td>
<td>111</td>
</tr>
<tr>
<td>Command Exit for NetView Commands</td>
<td>111</td>
</tr>
<tr>
<td>Writing C Exits</td>
<td>112</td>
</tr>
<tr>
<td>Building C Command Exit Programs</td>
<td>112</td>
</tr>
<tr>
<td>Installing C Programs as Command Exits</td>
<td>113</td>
</tr>
<tr>
<td>C Parameter Block (EGVE_Parameters32 Structure)</td>
<td>114</td>
</tr>
<tr>
<td>Command Exit Functions for C</td>
<td>123</td>
</tr>
<tr>
<td>IhsilInit - Register a Command Exit</td>
<td>123</td>
</tr>
<tr>
<td>IhsilInitialize - Register a Command Exit</td>
<td>124</td>
</tr>
<tr>
<td>IhsiSend - Remotely Invoke a Command Exit</td>
<td>124</td>
</tr>
<tr>
<td>IhsiTerminate - Unregister a Command Exit</td>
<td>125</td>
</tr>
<tr>
<td>IhsiWait - Wait Before Terminating</td>
<td>125</td>
</tr>
<tr>
<td>Return Codes</td>
<td>126</td>
</tr>
<tr>
<td>Return Codes from Command Exit Interface Functions</td>
<td>126</td>
</tr>
<tr>
<td>Other Return Codes</td>
<td>127</td>
</tr>
<tr>
<td>Invoking Command Exits with a C Interface</td>
<td>129</td>
</tr>
<tr>
<td>IHSDNATV Command Exit</td>
<td>129</td>
</tr>
<tr>
<td>IHSXTHCE Command Exit</td>
<td>130</td>
</tr>
<tr>
<td>IHSXTJAV Command Exit</td>
<td>132</td>
</tr>
<tr>
<td>IHSXTJCR Command Exit</td>
<td>133</td>
</tr>
<tr>
<td>Determining Additional Resource Information</td>
<td>134</td>
</tr>
<tr>
<td>Resource Manager Determination</td>
<td>134</td>
</tr>
<tr>
<td>Real or Aggregate Determination</td>
<td>134</td>
</tr>
</tbody>
</table>
Chapter 9. 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 NetView management console databases
- NetView features available through NetView management console

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 81.

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.

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 z/OS environment:

1. Open a workstation command window or shell prompt, depending on your operating system.
2. Change to one of the following directories:
   - For Windows: %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 both Windows and UNIX, if the Tivoli Framework is not installed on the platform 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 Windows, AIX, and Linux for zSeries™, 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 AIX, aix4-r1
  – For Linux for zSeries linux-s390

3. Enter one of the following commands to start the topology server:
   • For Windows: tserver start
   • For AIX: ./tserver start
   • For Linux for zSeries: ./tserver start

Because Linux for zSeries has no display, perform the following steps before
issuing this command to start the server:
   a. Telnet to the host.
   b. From the workstation used to perform the telnet, export the display to
      the workstation. To do this, use the following command:

         export DISPLAY=xxx.xx.xxx.xxx:0.0

      where xxx.xx.xxx.xxx is the dotted IP address of the workstation.

Notes:
   a. You must be the root user to start the topology server.
   b. On a UNIX system, if the topology server has been manually stopped and
      cannot be restarted, see Step [on page 9].

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 PORT=port_num
Where *topo_server_hostname* is the TCP/IP host name and *port_num* is the port number for the machine on which the topology server is running. The PORT parameter is optional.

- 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 z/OS environment.

  For more information about the NETCONV command, refer to the *Tivoli NetView for z/OS 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, or you can start it by issuing the command in line mode from any operating system, or from the Tivoli Desktop.

**Note:** You can also launch the NetView management console from other applications. For information, see "Appendix C. Launching and Using the NetView Management Console from Other Applications" on page 163.

### Selecting the Desktop Icon in Windows

For Windows, an icon is created on the desktop. To start the Topology Console using Windows, double-click the icon.

### Using a Line Command

Change to the appropriate directory for your environment:

- For Windows: `\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 NT and Windows 2000: `tconsoleNT .. -key nmc`
- For UNIX: `tconsole.sh .. -key nmc`

See "Appendix B. Topology Console Commands" on page 157 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 Topology Console Sign On Window

Figure 18 on page 82 shows the topology console Sign On window.
The following steps describe how to sign on to the topology server.

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

Notes:

a. The user ID must match your NetView user ID.

b. To restrict access to selected NetView operators to sign on to the topology console, use the LOGON command with the NMC keyword. This can be protected by command security (NetView command authorization table or
SAF), and permitted to particular operators or groups. An example of this is commented out in the sample command authorization table CNMSCAT2 and the sample SAF command authorization table CNMSAF2. For more information, refer to the Tivoli NetView for z/OS Security Reference.

2. Enter a password in the Password entry field.
   This password must match your NetView for z/OS 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 80 for more information.

4. If you have previously signed onto NetView management console 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 Tivoli NetView for z/OS Administration Reference book. Specifically, see the NGMFADMN keyword.

6. Select OK to sign on to NetView management console. The topology console window, as shown in Figure 19 on page 85 is displayed.

   Note: The first time you start the NetView management console, expect a delay as support files are downloaded to the topology console. On subsequent invocations of the topology console, only files which have been updated on the topology server are downloaded.

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 Tivoli NetView for z/OS Administration Reference book.

2. For more information about how files are automatically downloaded from the server at sign on, see “Appendix F. Automatic File Download at Console Log On” on page 197.

The Topology Console Window

Figure 19 on page 85 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 17 on page 36 and "The Filter Bar" on page 86 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
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.

Note: The filter bar is not displayed by default. To display the filter bar, select Options → Show View Filter.

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.

NetView Management Console Online Help

NetView management console 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.

Note: If you receive messages about the topology server, look in the topology console Help Index for information about the messages.

NetView Management Console Features and Functions

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

Issuing IP Commands

The following IP commands can be issued from the NetView management console interfaces:

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

SNMP commands such as GET, SET, GETNEXT, GETBULK, WALK, BULKWALK, and remote ping.

You can issue the available IP commands from the NMC topology console screen in one of two ways:

- From a specific resource, right-click the resource and select **IP Commands** 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 the pop-up menu.

From the **IP Commands** menu, the following items are available:

- onetstat
- oping
- otracert
- MIB Browser
- Real Time Poller
- SNMP Commands

When you select oping, onetstat, or otracert, a dialog is displayed. If the menu was accessed from a resource and you select oping or otracert, the IP address for that resource is inserted into the Host Machine field of the dialog. The oping command is also available to be started without opening a dialog when accessed from a resource. For more information on these dialogs and the options you can specify, refer to the NetView management console online help.

When you select MIB Browser, Real Time Poller, or SNMP Commands, the Web browser is launched with the Web address for the selected item. If the item is selected from a resource, the IP address for the resource is passed to the web application. To enable this function, the address of the web server to be accessed (either WebSphere® or Jetty) must be listed in the CNMSTYLE file. Refer to the `zNetViewWebApp_en.html` readme file for instructions on installing and customizing your Web application server environment.

**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. 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 domains. This view of NetView domains 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.

<table>
<thead>
<tr>
<th>If you want information about...</th>
<th>Refer to...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sample views</td>
<td>Tivoli NetView for z/OS Resource Object Data Manager and GMFHS Programmer’s Guide</td>
</tr>
</tbody>
</table>

Using the NetView Resource Manager

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 20 shows the RODM Collection Manager main menu.

![RODM Collection Manager](image)

Figure 20. The RODM Collection Manager main menu
Writing Server Information to the Topology Server Databases

Topology server databases are available in the directories shown in Table 49.

Table 49. Location of topology server databases

<table>
<thead>
<tr>
<th>Database</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default databases</td>
<td>For Windows: %BINDIR%\TDS\server\db\default\data</td>
</tr>
<tr>
<td></td>
<td>For UNIX: $BINDIR/TDS/server/db/default/data</td>
</tr>
<tr>
<td>Current databases</td>
<td>For Windows: %BINDIR%\TDS\server\db\current\data</td>
</tr>
<tr>
<td>(Working set of databases)</td>
<td>For UNIX: $BINDIR/TDS/server/db/current/data</td>
</tr>
<tr>
<td>Backup databases</td>
<td>For Windows: %BINDIR%\TDS\server\db\backup\data</td>
</tr>
<tr>
<td>(Backup of the current databases)</td>
<td>For UNIX: $BINDIR/TDS/server/db/backup/data</td>
</tr>
</tbody>
</table>

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" on page 91.

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" 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 79 for more information.

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.

Stopping the Topology Server

The procedure for stopping the topology server depends on the platform on which the topology server is running. For Windows, you can stop the topology server by selecting an icon or entering a line mode command. For AIX, 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. For Linux for zSeries, you can stop the topology server one of two ways: with a line mode command or by clicking the X in an Xwindow if the server was started with tserver start.

Attention: Do not stop the topology server by clicking the X in the top-right corner of the window unless the topology server is running on the Linux for zSeries platform and was started with tserver start.

With the Service Version on Windows

To stop the topology server in this environment, select the Control Panel → Services control applet, or enter the ihxsrv 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 Windows: $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.
Chapter 10. Using the NetView Management Console
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.

NetView management console provides a command profile editor batch-utility on both Windows and UNIX platforms.

This chapter describes the command profile editor. For information about using the command profile editor batch-utility, see “Using the Command Profile Editor Batch Utility” on page 93.

Understanding the Command Profile Editor

The Command Profile Editor contains the following objects:

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

Resource Manager 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 z/OS Data Model Reference for a detailed description of these command indicators.

Commands and Command Set 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 which are responsible for carrying out the command

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

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 defining a page for Tivoli Enterprise Console resources.

**Profile 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 component description files or business description files (CDF/BDF) 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 NetView management console. You can use this to provide security for a topology server.

**Using the Command Profile Editor Batch Utility**

The command profile editor provides a batch utility program to add, change, or delete commands in the command profile editor database. The batch utility runs on the Windows and UNIX platforms supported by the topology server. The utility uses a text file (the response file) that contains responses to all the parameters used in creating commands.

Note: NetView management console provides a utility to convert NetView Graphic Monitor Facility (NGMF) command tree facility definition files to command profile editor response files. See “Appendix G. Converting NGMF Command Sets” on page 199 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 NMC topology server. The samples are provided to enable you to customize the commands. The file names are:

**ihsscpe**
Base commands that are shipped with the NMC topology server

**flccke** Commands for MultiSystem Manager (MSM) resources

These files are located as follows:

• ihsscpe, for Windows: %BINDIR%\TDS\server\sample\ihsscpe.xxx.rsp
• ihsscpe, for UNIX: $BINDIR/TDS/server/sample/ihsscpe.xxx.rsp
• flccke, for Windows: %BINDIR%\TDS\server\sample\flccke.xxx.rsp
• flccke, for UNIX: $BINDIR/TDS/server/sample/flccke.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 Windows: `%BINDIR%\TDS\server\bin`
- For UNIX: `$BINDIR/TDS/server/bin`

For more information about the `cpebatch` command, see "cpebatch" on page 141.

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:

```plaintext
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 Windows: `%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 50.

Table 50. Manager command block keywords

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Uniquely defines the manager to the command profile editor.</td>
</tr>
<tr>
<td>INDICATOR_HIGH</td>
<td>High end of the range of indicators controlled by this manager.</td>
</tr>
<tr>
<td>INDICATOR_LOW</td>
<td>Low end of the range of indicators controlled by the manager.</td>
</tr>
<tr>
<td>INDICATOR_nn</td>
<td>( 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+n, when ( x ) is the offset of this indicator from the low range.</td>
</tr>
<tr>
<td>COMMENT</td>
<td>Comment string. Limit of 256 characters.</td>
</tr>
</tbody>
</table>

**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 51.

Table 51. Command block keywords

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAME</td>
<td>Uniquely identifies the command to the command profile editor.</td>
</tr>
<tr>
<td>MENU_STRING</td>
<td>The string that is displayed on the context menu.</td>
</tr>
<tr>
<td>COMMENT</td>
<td>Comment string. Limit of 256 characters.</td>
</tr>
<tr>
<td>HTML_HELP_FILE</td>
<td>The name of the HTML file that contains the help for this command.</td>
</tr>
</tbody>
</table>
Table 51. Command block keywords (continued)

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTML_HELP_ANCHOR</td>
<td>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 NMC topology console, # is inserted, if necessary.</td>
</tr>
<tr>
<td>MIN_RESOURCES</td>
<td>Minimum number of resources that must be selected in a view before this command is enabled. The range is 1 to 10.</td>
</tr>
<tr>
<td>MAX_RESOURCES</td>
<td>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.</td>
</tr>
<tr>
<td>PAGE</td>
<td>Defines the characteristics of a page for an individual command. See Page Keywords in the Command Block for the items you can specify here.</td>
</tr>
<tr>
<td>RESOURCE_INDEP</td>
<td>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.</td>
</tr>
<tr>
<td>VERIFY</td>
<td>YES or NO. Indicates whether to issue a confirmation message before the command is sent to the manager.</td>
</tr>
</tbody>
</table>

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 52 on page 99.
Table 52. Page command block keywords

<table>
<thead>
<tr>
<th>Page Characteristic</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLIENT_PLATFORM_LIST</td>
<td>The topology console platform or platforms associated with this command page.</td>
</tr>
<tr>
<td></td>
<td>• The syntax for one platform is:</td>
</tr>
<tr>
<td></td>
<td>CLIENT_PLATFORM_LIST = platform</td>
</tr>
<tr>
<td></td>
<td>Where platform is the platform you are using. Valid platforms are: AIX, HP-UX, Solaris, Linux, Windows 2000, and Windows NT.</td>
</tr>
<tr>
<td></td>
<td>• The syntax for all platforms is: CLIENT_PLATFORM_LIST = GENERIC</td>
</tr>
<tr>
<td></td>
<td>• The syntax for two or more platforms is: CLIENT_PLATFORM_LIST = (</td>
</tr>
<tr>
<td></td>
<td>CLIENT_PLATFORM = platform1,platform2</td>
</tr>
<tr>
<td></td>
<td>CLIENT_PLATFORM = platform3,platform4</td>
</tr>
<tr>
<td></td>
<td>)</td>
</tr>
<tr>
<td></td>
<td>Where platform1, platform2, platform3, platform4, and so on, are the platforms you are using.</td>
</tr>
<tr>
<td></td>
<td>Note: When specifying more than one platform you must separate each platform with a comma.</td>
</tr>
<tr>
<td></td>
<td>The topology console must be running on the specified platform to enable the command.</td>
</tr>
<tr>
<td></td>
<td>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.</td>
</tr>
</tbody>
</table>
Table 52. Page command block keywords (continued)

<table>
<thead>
<tr>
<th>Page Characteristic</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TARGET_PLATFORM_LIST</td>
<td>The target platform or platforms where the selected resource is located.</td>
</tr>
<tr>
<td></td>
<td>• The syntax for one platform is:</td>
</tr>
<tr>
<td></td>
<td>TARGET_PLATFORM_LIST = platform</td>
</tr>
<tr>
<td></td>
<td>Where <em>platform</em> is the platform you are using. You can specify: unknown, Other, sunos4, solaris2, hpux9, hpux10, aix3-r2, aix4, winNT, netware3, netware4, or NetView/390. Other platforms can be specified and will be added to the command profile editor list of valid platforms.</td>
</tr>
<tr>
<td></td>
<td>• The syntax for all platforms is:</td>
</tr>
<tr>
<td></td>
<td>TARGET_PLATFORM_LIST = GENERIC</td>
</tr>
<tr>
<td></td>
<td>• The syntax for two or more platforms is:</td>
</tr>
<tr>
<td></td>
<td>TARGET_PLATFORM_LIST = (</td>
</tr>
<tr>
<td></td>
<td>TARGET_PLATFORM = platform1,platform2</td>
</tr>
<tr>
<td></td>
<td>TARGET_PLATFORM = platform3,platform4</td>
</tr>
<tr>
<td></td>
<td>)</td>
</tr>
<tr>
<td></td>
<td>Where <em>platform1</em>, <em>platform2</em>, <em>platform3</em>, and <em>platform4</em>, and so on, are the platforms you are using.</td>
</tr>
<tr>
<td></td>
<td><strong>Note</strong>: When specifying more than one platform, separate each platform with a comma.</td>
</tr>
<tr>
<td></td>
<td>The target must be running on the specified platform to enable the command.</td>
</tr>
<tr>
<td></td>
<td><strong>Note</strong>: 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.</td>
</tr>
<tr>
<td>MANAGER_NAME</td>
<td>The name of the manager to which the page applies.</td>
</tr>
<tr>
<td></td>
<td>This must be previously defined in the response file.</td>
</tr>
<tr>
<td></td>
<td>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.</td>
</tr>
<tr>
<td>PAGE_ID</td>
<td>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.</td>
</tr>
</tbody>
</table>
Table 52. Page command block keywords (continued)

<table>
<thead>
<tr>
<th>Page Characteristic</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>INDICATOR_LIST</td>
<td>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 )</td>
</tr>
<tr>
<td>COMMAND_LIST</td>
<td>The command string to be sent to the command exit specified by EXIT_NAME.</td>
</tr>
<tr>
<td>EXIT_NAME</td>
<td>The name of the command exit to invoke for this page. See “Using Topology Server Command Exits” on page 106 for information.</td>
</tr>
<tr>
<td>LU_NAME</td>
<td>Currently, this field is ignored by NetView management console. All exits are executed on the topology server workstation.</td>
</tr>
<tr>
<td>HTML_HELP_FILE</td>
<td>The name of the HTML file that contains the help for this command.</td>
</tr>
<tr>
<td>HTML_HELP_ANCHOR</td>
<td>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.</td>
</tr>
<tr>
<td>IP_RESOURCE</td>
<td>IP Resource commands are displayed in the NetView management console when a resource with an IP address is right-clicked. Valid values are YES or NO. The default value is NO.</td>
</tr>
</tbody>
</table>

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 53.

Table 53. Command set block keywords

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAME</td>
<td>Uniquely identifies the command set to the command profile editor.</td>
</tr>
<tr>
<td>COMMENT</td>
<td>Comment string. Limit of 256 characters.</td>
</tr>
<tr>
<td>MENU_STRING</td>
<td>The string that is displayed on the Context menu.</td>
</tr>
<tr>
<td>HTML_HELP_FILE</td>
<td>The name of the HTML file that contains the help for this command set. This file must be installed on the topology server.</td>
</tr>
</tbody>
</table>
Table 53. Command set block keywords (continued)

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTML_HELP_ANCHOR</td>
<td>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.</td>
</tr>
<tr>
<td>MENU</td>
<td>Defines the commands that this command set contains and their order. You can specify one or more of the following:</td>
</tr>
<tr>
<td></td>
<td>• COMMAND_NAME - Specifies the name of a command to add.</td>
</tr>
<tr>
<td></td>
<td>• COMMAND_SET_NAME - Specifies the name of a command set to add.</td>
</tr>
<tr>
<td></td>
<td>• SEPARATOR - Specifies that a separator should be placed on the menu.</td>
</tr>
<tr>
<td></td>
<td>• REMOVE_COMMAND - Specifies the name of a command to remove.</td>
</tr>
<tr>
<td></td>
<td>• REMOVE_COMMAND_SET - Specifies the name of a command set to remove.</td>
</tr>
</tbody>
</table>

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:

```plaintext
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:

```plaintext
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:

```plaintext
COMMAND_SET = (  
    NAME = Network (identifies the command set)  
)```

102  NetView Management Console User’s Guide
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 54.

Table 54. Profile command block keywords

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

Operator Keywords
The operator block defines operators to the command database. The keywords used in the operator command block are shown in Table 55.

Table 55. Operator command block keywords

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAME</td>
<td>Uniquely defines the operator to the command profile editor</td>
</tr>
<tr>
<td>COMMENT</td>
<td>Comment string. Limit of 256 characters</td>
</tr>
<tr>
<td>PROFILE_NAME</td>
<td>The name of the profile that this operator will use</td>
</tr>
</tbody>
</table>
Chapter 11. 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.

Note: Topology server command exits are only supported on the Windows and AIX platforms.

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 NetView management console. 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 106. 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 110. 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 the TESTEXIT.EXE (for Windows platforms) or TESTEXIT (for UNIX platforms). This program registers 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 56.

<table>
<thead>
<tr>
<th>Command exit</th>
<th>Use to...</th>
<th>For information about...</th>
</tr>
</thead>
<tbody>
<tr>
<td>IHSDGENE</td>
<td>Send a fixed set of generic commands to the NetView Graphic Monitor Facility host subsystem (GMFHS) for processing.</td>
<td>• Using with the command profile editor, see <a href="#">IHSDGENE Command Exit</a></td>
</tr>
<tr>
<td>IHSDNATV</td>
<td>Send commands to GMFHS, which forwards the command to the service point for the specified resource.</td>
<td>• Using with the command profile editor, see <a href="#">IHSDNATV Command Exit</a> on page 107</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Invoking through a user-written command exit, see <a href="#">IHSDNATV Command Exit</a> on page 129</td>
</tr>
<tr>
<td>IHSXTHCE</td>
<td>Send a command to the NetView host.</td>
<td>• Using with the command profile editor, see <a href="#">IHSXTHCE Command Exit</a> on page 107</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Invoking through a user-written command exit, see <a href="#">IHSXTHCE Command Exit</a> on page 130</td>
</tr>
<tr>
<td>IHSXTJAM</td>
<td>Launch a single Java class on the topology console for multiple resources.</td>
<td>• Using with the command profile editor, see <a href="#">IHSXTJAM Command Exit</a> on page 108</td>
</tr>
<tr>
<td>IHSXTJAV</td>
<td>Start a Java class on the topology console.</td>
<td>• Using with the command profile editor, see <a href="#">IHSXTJAV Command Exit</a> on page 108</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Invoking through a user-written command exit, see <a href="#">IHSXTJAV Command Exit</a> on page 132</td>
</tr>
</tbody>
</table>

### 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 fllcpe.xxx.rsp response files, where xxx is a country code indicator, such as *en_US*.  

---

106 NetView Management Console User’s Guide
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 z/OS 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 z/OS. If you selected *Resource dependent*, you can optionally specify substitution variables in the Command string field. See "Substitution Variables" on page 108 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.
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.

Directs the command response back to the NetView for z/OS session.

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.

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 7. Topology Console Java Applications and Plug-ins" on page 59 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 124 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 7. Topology Console Java Applications and Plug-ins" on page 59 for information about installing this Java class.

Substitution Variables

Table 57 on page 109 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 57. Common substitution variables

<table>
<thead>
<tr>
<th>Substitution Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>%data1%</code></td>
<td>Substitutes the data1 from EGVE_PARAMETERS32.</td>
</tr>
<tr>
<td><code>%data2%</code></td>
<td>Substitutes the data2 from EGVE_PARAMETERS32.</td>
</tr>
<tr>
<td><code>%data3%</code></td>
<td>Substitutes the data3 from EGVE_PARAMETERS32.</td>
</tr>
<tr>
<td><code>%data4%</code></td>
<td>Substitutes the data4 from EGVE_PARAMETERS32.</td>
</tr>
<tr>
<td><code>%hb_hostname%</code></td>
<td>Substitutes the hb_hostname from EGVE_PARAMETERS32. Valid only for instrumented resources.</td>
</tr>
<tr>
<td><code>%hb_origin%</code></td>
<td>Substitutes the hb_origin from EGVE_PARAMETERS32. Valid only for instrumented resources.</td>
</tr>
<tr>
<td><code>%hb_primary%</code></td>
<td>First key value pair of hb_origin field from EGVE_PARAMETERS32. Valid only for instrumented resources.</td>
</tr>
<tr>
<td><code>%hb_secondary%</code></td>
<td>First key value pair of hb_sub_origin field from EGVE_PARAMETERS32. Valid only for instrumented resources.</td>
</tr>
<tr>
<td><code>%hb_source%</code></td>
<td>Substitutes the hb_source from EGVE_PARAMETERS32. Valid only for instrumented resources.</td>
</tr>
<tr>
<td><code>%hb_sub_origin%</code></td>
<td>Substitutes the hb_sub_origin from EGVE_PARAMETERS32. Valid only for instrumented resources.</td>
</tr>
<tr>
<td><code>%hb_sub_source%</code></td>
<td>Substitutes the hb_sub_source from EGVE_PARAMETERS32. Valid only for instrumented resources.</td>
</tr>
<tr>
<td><code>%ipaddress%</code></td>
<td>Substitutions data3 from EGVE_PARAMETERS32. Valid only for RODM resources.</td>
</tr>
<tr>
<td><code>%label%</code></td>
<td>Substitutes the display_name from EGVE_PARAMETERS32.</td>
</tr>
<tr>
<td><code>%monitor%</code></td>
<td>Substitutes the monitor name from the topology server database. Valid only for instrumented resources.</td>
</tr>
<tr>
<td><code>%objectid%</code></td>
<td>Substitutes resource_RODM_id in EGVE_PARAMETERS32.</td>
</tr>
<tr>
<td><code>%remoteconsole%</code></td>
<td>See <a href="#">%REMOTECONSOLE%</a> for more information.</td>
</tr>
<tr>
<td><code>%RODM.xxx%</code></td>
<td>Substitutes the variable to be pulled from RODM. The character string following the period (xxx) is the field name to be pulled from RODM for the selected object. These should be used only for resource specific commands.</td>
</tr>
<tr>
<td></td>
<td>To extract just a period-delimited token from a field, append a colon (:) and a token number as follows: %RODM.xxx:3%</td>
</tr>
<tr>
<td></td>
<td>This extracts the third period-delimited token from the field.</td>
</tr>
<tr>
<td><code>%tme_oid%</code></td>
<td>Substitutes the resource_TME_oid from EGVE_PARAMETERS32. Valid only for instrumented resources.</td>
</tr>
</tbody>
</table>

**%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\ihsscape.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 8. Configuring Property Files for Locally Launched Applications” on page 69 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 58.

Table 58. Command exits that can be invoked by user-written commands

<table>
<thead>
<tr>
<th>Command exit</th>
<th>Use to...</th>
<th>For information about...</th>
</tr>
</thead>
<tbody>
<tr>
<td>IHSXTHCE</td>
<td>Send a command to 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.</td>
<td>• Invoking through a user-written command exit, see [IHSXTHCE Command Exit on page 131]&lt;br&gt;• Using with the command profile editor, see [IHSXTHCE Command Exit on page 107]</td>
</tr>
<tr>
<td>IHSXTHCE</td>
<td>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.</td>
<td>• Invoking through a user-written command exit, see [IHSXTHCE Command Exit on page 131]&lt;br&gt;• Using with the command profile editor, see [IHSXTHCE Command Exit on page 107]</td>
</tr>
</tbody>
</table>
Table 58. Command exits that can be invoked by user-written commands (continued)

<table>
<thead>
<tr>
<th>Command exit</th>
<th>Use to...</th>
<th>For information about...</th>
</tr>
</thead>
</table>
| 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 on page 132  
• Using with the command profile editor, see IHSXTJAV on page 108 |
| 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 on page 133 |

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 21 on page 112 illustrates the sequence for invoking a typical command exit for NetView commands:
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 ihsixit.h header file. This file defines the relevant command exit data structures and function prototypes and is part of the topology server. The ihsixit.lib library must also be specified when linking.

The ihsixit.h header file is installed in one of the following:

- For Windows: %BINDIR%\TDS\server\sample directory
- For UNIX: $BINDIR/TDS/server/sample directory

Figure 21. Components and Events Involved in Command Exit Invocation
The ihsiexit.lib file is installed in one of the following:

- For Windows: %BINDIR%\TDS\server\lib directory
- For UNIX: $BINDIR/TDS/server/lib directory

The topology server provides the exit ihssame.c sample command in the following directories:

- For Windows: %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:

```c
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 a Windows 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 10. Using the NetView Management Console Command Profile Editor" on page 93.

2. Use the application programming interface, described in "Command Exit Functions for C" on page 123, 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 114.
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 Windows: `%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 59 describes the EGVE_PARAMETERS32 fields.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
<th>Type (Size)</th>
</tr>
</thead>
<tbody>
<tr>
<td>business_sys_name</td>
<td>Resource business system name.</td>
<td>PSZ (4 bytes)</td>
</tr>
<tr>
<td>client_handle</td>
<td>The IBM handle of the topology console that issued the command.</td>
<td>ULONG (4 bytes)</td>
</tr>
<tr>
<td>client_hostname</td>
<td>The TCP hostname of the topology console that is invoking the command.</td>
<td>PSZ (4 bytes)</td>
</tr>
<tr>
<td>client_ip_addr</td>
<td>The address of the topology console invoking this command.</td>
<td>SZ (9 bytes)</td>
</tr>
<tr>
<td></td>
<td>This is an IBM internal version of the socket file descriptor associated with the server's IP connection with the topology console.</td>
<td></td>
</tr>
<tr>
<td>client_ip_addr_dd</td>
<td>IP dotted decimal address of the topology console that is invoking the command.</td>
<td>PSZ (4 bytes)</td>
</tr>
<tr>
<td>command_indicator</td>
<td>This field determines the command page used in command profile editor to execute this command.</td>
<td>USHORT (2 bytes)</td>
</tr>
<tr>
<td>command_name</td>
<td>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.</td>
<td>PSZ (4 bytes)</td>
</tr>
<tr>
<td>command_string</td>
<td>The command string.</td>
<td>PSZ (4 bytes)</td>
</tr>
<tr>
<td>Field</td>
<td>Description</td>
<td>Type (Size)</td>
</tr>
<tr>
<td>------------------</td>
<td>------------------------------------------------------------------------------</td>
<td>-------------------</td>
</tr>
<tr>
<td>correlation_id</td>
<td>A unique number used to correlate requests and responses. This field in the</td>
<td>struct</td>
</tr>
<tr>
<td></td>
<td>EGVE_PARAMETERS32 structure should not be modified by a user command exit.</td>
<td>EGV_CORR_ID (4</td>
</tr>
<tr>
<td></td>
<td>Altering this field could result in an error when this exit returns to its</td>
<td>bytes)</td>
</tr>
<tr>
<td></td>
<td>invoker. This number is used by the topology server to correlate the command</td>
<td></td>
</tr>
<tr>
<td></td>
<td>to its command responses. The command string is sent by the topology server</td>
<td></td>
</tr>
<tr>
<td></td>
<td>to the topology console if you set want_corr_of_cmd_rsps before returning</td>
<td></td>
</tr>
<tr>
<td></td>
<td>from your registered exit procedure. Then you can send command responses</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(using the IhsiSend API) to your command by calling the IHSX_TJR command</td>
<td></td>
</tr>
<tr>
<td></td>
<td>exit using the same correlation number. If your exit is calling another exit</td>
<td></td>
</tr>
<tr>
<td></td>
<td>and you create a new instance of the EGVE_PARAMETERS32 block, copy the</td>
<td></td>
</tr>
<tr>
<td></td>
<td>correlation_id passed from the caller into the new instance.</td>
<td></td>
</tr>
<tr>
<td>data1</td>
<td>Resource specific data1. For RODM resources, this data is from the Display</td>
<td>PSZ (4 bytes)</td>
</tr>
<tr>
<td></td>
<td>ResourceOtherData field. For RODM predefined views in the tree view, this</td>
<td></td>
</tr>
<tr>
<td></td>
<td>data is from the Annotation field.</td>
<td></td>
</tr>
<tr>
<td>data2</td>
<td>Resource specific data2. For RODM resources, this data is from the Display</td>
<td>PSZ (4 bytes)</td>
</tr>
<tr>
<td></td>
<td>ResourceUserData field.</td>
<td></td>
</tr>
<tr>
<td>data3</td>
<td>Resource specific data3. For RODM resources, this data is from the ipAddress</td>
<td>PSZ (4 bytes)</td>
</tr>
<tr>
<td></td>
<td>field.</td>
<td></td>
</tr>
<tr>
<td>data4</td>
<td>Resource specific data4.</td>
<td>PSZ (4 bytes)</td>
</tr>
<tr>
<td>data_source</td>
<td>This field indicates the source of the resource. The possible values are:</td>
<td>USHORT (2 bytes)</td>
</tr>
<tr>
<td></td>
<td>• 0 - All RODM resources, except for SNA Shadow objects and SNA Topology</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Manager resources</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• 1 - RODM SNA Shadow objects (from the GMFHS_Shadow_Objects_Class)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• 2 - Topology Display Manager resources (Business System resources managed</td>
<td></td>
</tr>
<tr>
<td></td>
<td>by the topology server)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• 5 - RODM SNA Topology Manager resources</td>
<td></td>
</tr>
<tr>
<td>display_name</td>
<td>The resource name assigned by the manager of the resource. This is the name</td>
<td>PSZ (4 bytes)</td>
</tr>
<tr>
<td></td>
<td>that appears in the view. For RODM resources, this data is from the Display</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ResourceName field.</td>
<td></td>
</tr>
<tr>
<td>Field</td>
<td>Description</td>
<td>Type (Size)</td>
</tr>
<tr>
<td>---------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>-------------</td>
</tr>
</tbody>
</table>
| 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=\textit{The managed node name of the event server to which instrumentation sent a heartbeat}  
  This is not filled in for NetView instrumentation.  
  • 2=\textit{The managed node name of the event server by which istetec was driven}  
  This is always filled in.  
An example of this field is:  
1=kid.gt.com;2=pup.dg.org | PSZ (4 bytes) |

Does not apply to RODM resources. Only available for instrumented resources.
Table 59. The EGVE_PARAMETERS32 fields (continued)

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
<th>Type (Size)</th>
</tr>
</thead>
<tbody>
<tr>
<td>hb_hostname</td>
<td>In the distributed environment, the value must be the TCP/IP host name of the system. In the z/OS 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.</td>
<td>PSZ (4 bytes)</td>
</tr>
<tr>
<td>hb_origin</td>
<td>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.</td>
<td>PSZ (4 bytes)</td>
</tr>
<tr>
<td>hb_source</td>
<td>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.</td>
<td>PSZ (4 bytes)</td>
</tr>
<tr>
<td>hb_sub_origin</td>
<td>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.</td>
<td>PSZ (4 bytes)</td>
</tr>
<tr>
<td>hb_sub_source</td>
<td>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.</td>
<td>PSZ (4 bytes)</td>
</tr>
<tr>
<td>IBM_data</td>
<td>Tivoli use only.</td>
<td>PBYTE (4 bytes)</td>
</tr>
<tr>
<td>IBM_data_length</td>
<td>The length of the Tivoli use only data.</td>
<td>USHORT (2 bytes)</td>
</tr>
<tr>
<td>IBM_menu_id</td>
<td>The Tivoli internal command profile editor menu ID.</td>
<td>ULONG (4 bytes)</td>
</tr>
<tr>
<td>nv390_domain</td>
<td>NetView domain ID.</td>
<td>SZ (6 bytes)</td>
</tr>
<tr>
<td></td>
<td>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.</td>
<td></td>
</tr>
<tr>
<td>nv390_hostname</td>
<td>TCP hostname of the NetView host, if available.</td>
<td>PSZ (4 bytes)</td>
</tr>
<tr>
<td>nv390_ip_addr</td>
<td>IP dotted decimal address of the NetView host, if available.</td>
<td>PSZ (4 bytes)</td>
</tr>
</tbody>
</table>
Table 59. The EGVE_PARAMETERS32 fields (continued)

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
<th>Type (Size)</th>
</tr>
</thead>
<tbody>
<tr>
<td>op_id</td>
<td>The NetView operator ID, if available.</td>
<td>SZ (9 bytes)</td>
</tr>
<tr>
<td>res_IBM_id</td>
<td>The topology server ID, for the resource. This field contains a zero for a</td>
<td>ULONG (4 bytes)</td>
</tr>
<tr>
<td></td>
<td>resource independent command.</td>
<td></td>
</tr>
<tr>
<td>resource_flags</td>
<td>The flags of the selected resource. Zero, one, or more values can be set at</td>
<td>ULONG (4 bytes)</td>
</tr>
<tr>
<td></td>
<td>one time. Possible values are:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x8000000000</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Marked</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x4000000000</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Suspended</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x2000000000</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Automatically clear suspended</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x1000000000</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SNA alert pending</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x0800000000</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Threshold inconsistency</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x0400000000</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Automation in progress</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x0200000000</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Not monitored</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x0100000000</td>
<td></td>
</tr>
<tr>
<td></td>
<td>IBM reserved 1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x0080000000</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Child suspended</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x0040000000</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Status not valid</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x0020000000</td>
<td></td>
</tr>
<tr>
<td></td>
<td>IBM reserved 4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x0010000000</td>
<td></td>
</tr>
<tr>
<td></td>
<td>IBM reserved 5</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x0008000000</td>
<td></td>
</tr>
<tr>
<td></td>
<td>IBM reserved 6</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x0004000000</td>
<td></td>
</tr>
<tr>
<td></td>
<td>IBM reserved 7</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x0002000000</td>
<td></td>
</tr>
<tr>
<td></td>
<td>IBM reserved 8</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x0001000000</td>
<td></td>
</tr>
<tr>
<td></td>
<td>IBM reserved 9</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x0000800000</td>
<td></td>
</tr>
<tr>
<td></td>
<td>IBM reserved 10</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x0000400000</td>
<td></td>
</tr>
<tr>
<td></td>
<td>IBM reserved 11</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x0000200000</td>
<td></td>
</tr>
<tr>
<td></td>
<td>IBM reserved 12</td>
<td></td>
</tr>
</tbody>
</table>
Table 59. The EGVE_PARAMETERS32 fields (continued)

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
<th>Type (Size)</th>
</tr>
</thead>
<tbody>
<tr>
<td>resource_flags</td>
<td>The flags of the selected resource. Zero, one, or more values can be set at one time. Possible values are:</td>
<td></td>
</tr>
<tr>
<td>con't</td>
<td></td>
<td>ULONG (4 bytes)</td>
</tr>
<tr>
<td>resource_RODM_id</td>
<td>The NetView RODM object ID, if applicable.</td>
<td>CHAR (8 bytes)</td>
</tr>
<tr>
<td>resource_status</td>
<td>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.</td>
<td>UNSIGNED CHAR (1 byte)</td>
</tr>
<tr>
<td>resource_status_ts</td>
<td>This is the resource_status timestamp.</td>
<td>struct EGVE_TIMESTAMP (8 bytes)</td>
</tr>
<tr>
<td>resource_TME_oid</td>
<td>The TME® object ID, if available. Does not apply to RODM resources. Only available for instrumented resources.</td>
<td>PSZ (4 bytes)</td>
</tr>
<tr>
<td>resource_type</td>
<td>The identifier that uniquely defines the type of resource. For RODM resources, look in Display_Resource_Type_Class in RODM.</td>
<td>ULONG (4 bytes)</td>
</tr>
<tr>
<td>Field</td>
<td>Description</td>
<td>Type (Size)</td>
</tr>
<tr>
<td>------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>server_ip_addr</td>
<td>The IP address of the topology server workstation.</td>
<td>SZ (9 bytes)</td>
</tr>
<tr>
<td></td>
<td>The first 4 bytes contain '0xFF'. The second 4 bytes is the first address</td>
<td></td>
</tr>
<tr>
<td></td>
<td>in the address list of the host entry struct returned by gethostbyname().</td>
<td></td>
</tr>
<tr>
<td>signon_username</td>
<td>The user name as entered on the topology console Sign On window.</td>
<td>PSZ (4 bytes)</td>
</tr>
<tr>
<td>user_ctrl_data</td>
<td>A pointer to user-determined control data. This field is used for arbitrary</td>
<td>PBYTE (4 bytes)</td>
</tr>
<tr>
<td></td>
<td>exit-to-exit communication.</td>
<td></td>
</tr>
<tr>
<td>user_ctrl_data_len</td>
<td>The length of the user control data field.</td>
<td>USHORT (2 bytes)</td>
</tr>
<tr>
<td>user_data</td>
<td>A pointer to user-determined data. This field is used for arbitrary exit-to-exit communication.</td>
<td>PBYTE (4 bytes)</td>
</tr>
<tr>
<td>user_data_length</td>
<td>The length of the user data field.</td>
<td>USHORT (2 bytes)</td>
</tr>
<tr>
<td>user_res_data</td>
<td>User resource data. A pointer to resource data determined by the user. This field is used for arbitrary exit-to-exit communications.</td>
<td>PBYTE (4 bytes)</td>
</tr>
<tr>
<td>user_res_data_length</td>
<td>The length of the user_res_data field.</td>
<td>USHORT (2 bytes)</td>
</tr>
<tr>
<td>wait_for_cmd_response</td>
<td>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.</td>
<td>SHORT (2 bytes)</td>
</tr>
<tr>
<td>want_corr_of_cmd_rsp</td>
<td>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.</td>
<td>USHORT (2 bytes)</td>
</tr>
</tbody>
</table>

**Table 60. Definitions of the data types (with all strings in ASCII).**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHAR</td>
<td>A character field</td>
</tr>
<tr>
<td>LONG</td>
<td>32-bit signed integer</td>
</tr>
<tr>
<td>PBYTE</td>
<td>Pointer to binary data</td>
</tr>
<tr>
<td>PSZ</td>
<td>Pointer to null-terminated string</td>
</tr>
<tr>
<td>SHORT</td>
<td>16-bit signed integer</td>
</tr>
<tr>
<td>struct</td>
<td>Defined in ihsiexit.h</td>
</tr>
<tr>
<td>SZ</td>
<td>A null-terminated string</td>
</tr>
<tr>
<td>ULONG</td>
<td>32-bit unsigned integer</td>
</tr>
</tbody>
</table>
Table 60. Definitions of the data types (with all strings in ASCII). (continued)

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>USHORT</td>
<td>16-bit unsigned integer</td>
</tr>
</tbody>
</table>

Table 61 maps the resource status values returned in the resource_status field in EGVE_PARAMETERS32 to the descriptions for the RODM DisplayStatus status schemes.

Table 61. Descriptions of the resource_status values in EGVE_PARAMETERS32

<table>
<thead>
<tr>
<th>resource_status Value in EGVE_PARAMETERS32</th>
<th>RODM DisplayStatus Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Satisfactory</td>
</tr>
<tr>
<td>2</td>
<td>Medium satisfactory</td>
</tr>
<tr>
<td>5</td>
<td>Low satisfactory</td>
</tr>
<tr>
<td>6</td>
<td>Intermediate</td>
</tr>
<tr>
<td>9</td>
<td>Degraded</td>
</tr>
<tr>
<td>12</td>
<td>Low unsatisfactory</td>
</tr>
<tr>
<td>14</td>
<td>Severely degraded</td>
</tr>
<tr>
<td>17</td>
<td>Medium unsatisfactory</td>
</tr>
<tr>
<td>18</td>
<td>Unsatisfactory</td>
</tr>
<tr>
<td>20</td>
<td>Unknown</td>
</tr>
<tr>
<td>21</td>
<td>Deleted</td>
</tr>
<tr>
<td>24</td>
<td>Scheduled</td>
</tr>
<tr>
<td>25</td>
<td>_IBM positive 4</td>
</tr>
<tr>
<td>26</td>
<td>_IBM positive 5</td>
</tr>
<tr>
<td>27</td>
<td>_IBM positive 6</td>
</tr>
<tr>
<td>28</td>
<td>_IBM positive 7</td>
</tr>
<tr>
<td>29</td>
<td>_IBM positive 8</td>
</tr>
<tr>
<td>30</td>
<td>_IBM negative 3</td>
</tr>
<tr>
<td>31</td>
<td>_IBM negative 4</td>
</tr>
<tr>
<td>32</td>
<td>_IBM negative 5</td>
</tr>
<tr>
<td>33</td>
<td>_IBM negative 6</td>
</tr>
<tr>
<td>34</td>
<td>_IBM negative 7</td>
</tr>
<tr>
<td>35</td>
<td>_IBM negative 8</td>
</tr>
<tr>
<td>36</td>
<td>_User positive 1</td>
</tr>
<tr>
<td>38</td>
<td>_User positive 2</td>
</tr>
<tr>
<td>39</td>
<td>_User positive 3</td>
</tr>
<tr>
<td>40</td>
<td>_User positive 4</td>
</tr>
<tr>
<td>41</td>
<td>_User positive 5</td>
</tr>
<tr>
<td>42</td>
<td>_User positive 6</td>
</tr>
<tr>
<td>43</td>
<td>_User positive 7</td>
</tr>
<tr>
<td>44</td>
<td>_User positive 8</td>
</tr>
<tr>
<td>45</td>
<td>_User negative 1</td>
</tr>
</tbody>
</table>
**Table 61. Descriptions of the resource_status values in EGVE_PARAMETERS32 (continued)**

<table>
<thead>
<tr>
<th>resource_status Value in EGVE_PARAMETERS32</th>
<th>RDOM DisplayStatus Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>47</td>
<td>_User negative 2</td>
</tr>
<tr>
<td>48</td>
<td>_User negative 3</td>
</tr>
<tr>
<td>49</td>
<td>_User negative 4</td>
</tr>
<tr>
<td>50</td>
<td>_User negative 5</td>
</tr>
<tr>
<td>51</td>
<td>_User negative 6</td>
</tr>
<tr>
<td>52</td>
<td>_User negative 7</td>
</tr>
<tr>
<td>53</td>
<td>_User negative 8</td>
</tr>
</tbody>
</table>

**Table 62** 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 62. The RODM DisplayStatus field and the resource_status value in EGVE_PARAMETERS32**

<table>
<thead>
<tr>
<th>DisplayStatus Value in RDOM</th>
<th>resource_status Value in EGVE_PARAMETERS32</th>
</tr>
</thead>
<tbody>
<tr>
<td>129</td>
<td>0</td>
</tr>
<tr>
<td>130</td>
<td>18</td>
</tr>
<tr>
<td>131</td>
<td>6</td>
</tr>
<tr>
<td>132</td>
<td>20</td>
</tr>
<tr>
<td>133</td>
<td>9</td>
</tr>
<tr>
<td>134</td>
<td>14</td>
</tr>
<tr>
<td>135</td>
<td>21</td>
</tr>
<tr>
<td>136</td>
<td>36</td>
</tr>
<tr>
<td>137</td>
<td>38</td>
</tr>
<tr>
<td>138</td>
<td>39</td>
</tr>
<tr>
<td>139</td>
<td>40</td>
</tr>
<tr>
<td>140</td>
<td>41</td>
</tr>
<tr>
<td>141</td>
<td>42</td>
</tr>
<tr>
<td>142</td>
<td>43</td>
</tr>
<tr>
<td>143</td>
<td>44</td>
</tr>
<tr>
<td>144</td>
<td>2</td>
</tr>
<tr>
<td>145</td>
<td>5</td>
</tr>
<tr>
<td>146</td>
<td>24</td>
</tr>
<tr>
<td>147</td>
<td>25</td>
</tr>
<tr>
<td>148</td>
<td>26</td>
</tr>
<tr>
<td>149</td>
<td>27</td>
</tr>
<tr>
<td>150</td>
<td>28</td>
</tr>
<tr>
<td>151</td>
<td>29</td>
</tr>
<tr>
<td>152</td>
<td>45</td>
</tr>
</tbody>
</table>
Table 62. The RODM DisplayStatus field and the resource_status value in EGVE_PARAMETERS32 (continued)

<table>
<thead>
<tr>
<th>DisplayStatus Value in RODM</th>
<th>resource_status Value in EGVE_PARAMETERS32</th>
</tr>
</thead>
<tbody>
<tr>
<td>153</td>
<td>47</td>
</tr>
<tr>
<td>154</td>
<td>48</td>
</tr>
<tr>
<td>155</td>
<td>49</td>
</tr>
<tr>
<td>156</td>
<td>50</td>
</tr>
<tr>
<td>157</td>
<td>51</td>
</tr>
<tr>
<td>158</td>
<td>52</td>
</tr>
<tr>
<td>159</td>
<td>53</td>
</tr>
<tr>
<td>160</td>
<td>17</td>
</tr>
<tr>
<td>161</td>
<td>12</td>
</tr>
<tr>
<td>162</td>
<td>30</td>
</tr>
<tr>
<td>163</td>
<td>31</td>
</tr>
<tr>
<td>164</td>
<td>32</td>
</tr>
<tr>
<td>165</td>
<td>33</td>
</tr>
<tr>
<td>166</td>
<td>34</td>
</tr>
<tr>
<td>167</td>
<td>35</td>
</tr>
</tbody>
</table>

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.

Ihsilnit - 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:

```c
unsigned int Ihsilnit(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 126.

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 123 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 126.

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 126.

If IhsiSend is able to invoke the command exit, the return value is one of those listed in "Other Return Codes" on page 127. 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 126.

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:

```c
unsigned int IhsiWait(char * exit_name, int seconds);
```

Where:

- `exit_name` is a null-terminated string (maximum 8 characters) that specifies the name of the command exit to wait.
- `seconds` is 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>
<thead>
<tr>
<th>Return Value</th>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EGVE_OK</td>
<td>0x0000</td>
<td>The interface function completed successfully.</td>
</tr>
<tr>
<td>EGVE_ALREADY_STARTED</td>
<td>0x0001</td>
<td>The exit name specified is already registered.</td>
</tr>
<tr>
<td>EGVE_PARAMETER_CHECK</td>
<td>0x0002</td>
<td>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.</td>
</tr>
<tr>
<td>EGVE_OUT_OF_MEMORY</td>
<td>0x0003</td>
<td>The command exit facility could not obtain memory to invoke the exit.</td>
</tr>
<tr>
<td>EGVE_NOT_STARTED</td>
<td>0x0004</td>
<td>A command exit interface function was called before an IhsiInitialize or IhsiInit was performed.</td>
</tr>
<tr>
<td>EGVE_HANDLER_NOT_STARTED</td>
<td>0x0005</td>
<td>Topology communication server has not been started correctly or is not running.</td>
</tr>
<tr>
<td>EGVE_DATA_OVERFLOW</td>
<td>0x0006</td>
<td>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.</td>
</tr>
</tbody>
</table>
Table 63. Return codes from command exits IhsiInitialize, IhsInit, IhsTerminate, IhsiSend, and IhsiWait (continued)

<table>
<thead>
<tr>
<th>Return Value</th>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EGVE_INTERNAL_FAILURE</td>
<td>0x0007</td>
<td>An unrecoverable error occurred in a command exit interface function.</td>
</tr>
<tr>
<td>EGVE_SEM_TIMEOUT</td>
<td>0x0008</td>
<td>When invoking IhsiWait, the time specified has expired.</td>
</tr>
<tr>
<td>EGVE_SEM_TERMINATE</td>
<td>0x0009</td>
<td>When invoking IhsiWait, the topology communication server is shutting down.</td>
</tr>
</tbody>
</table>

Other Return Codes

Table 64 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 64. Return codes from command exits

<table>
<thead>
<tr>
<th>Return Value</th>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EGVE_EXIT_OK</td>
<td>0x0000</td>
<td>The command exit completed successfully.</td>
</tr>
<tr>
<td>EGVE_EXIT_INVALID_NV_DOMAIN</td>
<td>0x801A</td>
<td>The NetView domain ID contains invalid syntax. The domain ID must meet the following criteria.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• It must be from 1 to 5 characters in length.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• The first character must be alphabetic (either upper or lower case) or one of the following characters: @#$,.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• The remaining characters (second through fifth) must be alphabetic (either upper or lower case), numeric, or one of the following characters: @#$,.</td>
</tr>
<tr>
<td>EGVE_EXIT_NOT_LOADED</td>
<td>0x8001</td>
<td>The exit name, specified by the dest_exit_name parameter of the IhsiSend function, could not be loaded.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Possible causes:</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• The dest_exit_name executable file could not be found.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• An IhsiInitialize or IhsInit was not performed for dest_exit_name.</td>
</tr>
<tr>
<td>EGVE_EXIT_FAIL</td>
<td>0x8002</td>
<td>The command exit invoked was not able to complete its function due to an error.</td>
</tr>
<tr>
<td>EGVE_EXIT_NO_OPER</td>
<td>0x8004</td>
<td>The operator ID specified in the command exit parameter block is not signed on to the NetView management console or the NetView host.</td>
</tr>
<tr>
<td>Return Value</td>
<td>Code</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------------------</td>
<td>--------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>EGVE_EXIT_INVALID_CMD</td>
<td>0x8005</td>
<td>The command string specified in the command exit parameter block is not valid; for example, the command is too long.</td>
</tr>
<tr>
<td>EGVE_EXIT_HOST_DOWN</td>
<td>0x8006</td>
<td>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.</td>
</tr>
<tr>
<td>EGVE_EXIT_NO_DOMAINS</td>
<td>0x8007</td>
<td>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.</td>
</tr>
<tr>
<td>EGVE_EXIT_OUT_OF_MEMORY</td>
<td>0x8009</td>
<td>The command exit did not obtain memory to complete its function.</td>
</tr>
<tr>
<td>EGVE_EXIT_OUT_OF_RESOURCES</td>
<td>0x800a</td>
<td>The command exit facility could not obtain a resource to invoke a command exit; for example, a thread could not be started.</td>
</tr>
<tr>
<td>EGVE_EXIT_INVALID_DEFINITION</td>
<td>0x800b</td>
<td>The data passed to the command exit was either not valid or was inconsistent with the requirements of the exit. Possible errors are:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• The command string was not valid or not defined.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• The exit required resource information, but the exit was defined as resource-independent.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• The particular resource type passed to an exit is not supported by that exit.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• The command defined to CPE references a DDF file, but the DDF file could not be located in the DPATH environment variable.</td>
</tr>
<tr>
<td>EGVE_EXIT_TME_TASK_NOT_FOUND</td>
<td>0x800c</td>
<td>The TME task was not found. Only returned by the IHSMTTME command exit.</td>
</tr>
<tr>
<td>EGVE_EXIT_TME_CATCH_IN_MAIN</td>
<td>0x800d</td>
<td>An unexpected exception was thrown while invoking the TME task. Only returned by IHSMTTME command exit.</td>
</tr>
<tr>
<td>EGVE_EXIT_TME_INVALID_MANAGEDNODE_NAME</td>
<td>0x800e</td>
<td>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.</td>
</tr>
<tr>
<td>EGVE_EXIT_TME_INVALID_COMMAND_STRING</td>
<td>0x800f</td>
<td>The syntax of the command string input is not correct. It did not follow the format of taskname</td>
</tr>
<tr>
<td>EGVE_EXIT_TME_ERR_GET_ALL_RESOURCES</td>
<td>0x8010</td>
<td>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.</td>
</tr>
</tbody>
</table>
Table 64. Return codes from command exits (continued)

<table>
<thead>
<tr>
<th>Return Value</th>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EGVE_EXIT_TME_BAD_RETURN_FROM_TAS</td>
<td>0x8011</td>
<td>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.</td>
</tr>
<tr>
<td>EGVE_EXIT_TME_ERROR_DURING_TASK_EXEC</td>
<td>0x8012</td>
<td>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.</td>
</tr>
<tr>
<td>EGVE_EXIT_OPER_NOT_AUTH</td>
<td>0x8019</td>
<td>The operator ID specified in the command exit parameter block is not authorized to issue NetView commands.</td>
</tr>
<tr>
<td>EGVE_EXIT_USER</td>
<td>0x9000</td>
<td>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.</td>
</tr>
<tr>
<td>EGVE_EXIT_USER_MAX</td>
<td>0x9fff</td>
<td>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.</td>
</tr>
</tbody>
</table>

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 65 illustrates the C interface to the IHSDNATV command exit.

Table 65. C interface to command exit IHSDNATV

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
<th>Type (Size)</th>
</tr>
</thead>
<tbody>
<tr>
<td>client_handle</td>
<td>The client_handle field must be the same as the client_handle field in EGVE_PARAMETERS parameter block of the command exit invoking IHSDNATV.</td>
<td>ULONG (4 bytes)</td>
</tr>
</tbody>
</table>
Table 65. C interface to command exit IHSXDHATV (continued)

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
<th>Type (Size)</th>
</tr>
</thead>
<tbody>
<tr>
<td>client_ip_addr</td>
<td>Set this field to the same value as that passed to you in the EGVE_PARAMETERS32 structure when your exit was invoked.</td>
<td>SZ (9 bytes)</td>
</tr>
<tr>
<td>command_string</td>
<td>The command string to send to GMFHS.</td>
<td>PSZ (4 bytes)</td>
</tr>
<tr>
<td>correlation_id</td>
<td>Set this field to the same value that is passed to you in the EGVE_PARAMETER32 structure when your exit was invoked.</td>
<td>struct EGVE_CORR_ID (4 bytes)</td>
</tr>
<tr>
<td>display_name</td>
<td>Set this field to the same value that is passed to you in the EGVE_PARAMETER32 structure when your exit was invoked.</td>
<td>PSZ (4 bytes)</td>
</tr>
<tr>
<td>op_id</td>
<td>The operator ID that has signed on to the NetView host</td>
<td>SZ (9 bytes)</td>
</tr>
<tr>
<td>resource_RODM_id</td>
<td>This field is the 8 bytes of a RODM object ID.</td>
<td>CHAR (8 bytes)</td>
</tr>
<tr>
<td>resource_type</td>
<td>Set this field to the same value that is passed to you in the EGVE_PARAMETER32 structure when your exit was invoked.</td>
<td>ULONG (4 bytes)</td>
</tr>
<tr>
<td>server_ip_addr</td>
<td>Set this field to the same value as that passed to you in the EGVE_PARAMETERS32 structure when your exit was invoked.</td>
<td>SZ (9 bytes)</td>
</tr>
</tbody>
</table>

**Note:** For data type definitions, see Table 60 on page 120.

**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 107 and "Substitution Variables" on page 108 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 66 illustrates the C interface to the IHSXTHCE command exit.

Table 66. C interface to command exit IHSXTHCE

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
<th>Type (Size)</th>
</tr>
</thead>
<tbody>
<tr>
<td>client_ip_addr</td>
<td>Set this field to the same value as that passed to you in the EGVE_PARAMETERS32 structure when your exit was invoked.</td>
<td>SZ (9 bytes)</td>
</tr>
<tr>
<td>command_indicator</td>
<td>Set this field to the same value as that passed to you in the EGVE_PARAMETERS32 structure when your exit was invoked.</td>
<td>USHORT (2 bytes)</td>
</tr>
<tr>
<td>command_string</td>
<td>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.</td>
<td>PSZ (4 bytes)</td>
</tr>
<tr>
<td>correlation_id</td>
<td>Set this field to the same value as that passed to you in the EGVE_PARAMETERS32 structure when your exit was invoked.</td>
<td>struct EGVE_CORR_ID (4 bytes)</td>
</tr>
<tr>
<td>data_source</td>
<td>Set this field to the same value as that passed to you in the EGVE_PARAMETERS32 structure when your exit was invoked.</td>
<td>USHORT (2 bytes)</td>
</tr>
<tr>
<td>display_name</td>
<td>Set this field to the same value that is passed to you in the EGVE_PARAMETER32 structure when your exit was invoked.</td>
<td>PSZ (4 bytes)</td>
</tr>
<tr>
<td>nv390_domain</td>
<td>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).</td>
<td>SZ (6 bytes)</td>
</tr>
<tr>
<td>op_id</td>
<td>The operator ID that has signed on to the NetView host.</td>
<td>SZ (9 bytes)</td>
</tr>
</tbody>
</table>
### Table 66. C interface to command exit IHSXHCE (continued)

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
<th>Type (Size)</th>
</tr>
</thead>
<tbody>
<tr>
<td>res_IBM_id</td>
<td>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.</td>
<td>ULONG (4 bytes)</td>
</tr>
<tr>
<td>resource_RODM_id</td>
<td>Set this field to the same value that is passed to you in the EGVE_PARAMETER32 structure when your exit was invoked.</td>
<td>CHAR (8 bytes)</td>
</tr>
<tr>
<td>resource_status</td>
<td>Set this field to the same value that is passed to you in the EGVE_PARAMETER32 structure when your exit was invoked.</td>
<td>UNSIGNED CHAR (1 byte)</td>
</tr>
<tr>
<td>server_ip_addr</td>
<td>Set this field to the same value as that passed to you in the EGVE_PARAMETERS32 structure when your exit was invoked.</td>
<td>SZ (9 bytes)</td>
</tr>
<tr>
<td>user_data</td>
<td>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.</td>
<td>PBYTE (4 bytes)</td>
</tr>
<tr>
<td>user_data_length</td>
<td>The length of the storage pointed to by user_data.</td>
<td>USHORT (2 bytes)</td>
</tr>
<tr>
<td>wait_for_cmd_response</td>
<td>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.</td>
<td>SHORT (2 bytes)</td>
</tr>
</tbody>
</table>

**Note:** For data type definitions, see Table 60 on page 120.

### 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 7. Topology Console Java Applications and Plug-ins” on page 59 for information about installing these Java classes.
Table 67 illustrates the C interface to the IHSXTJAV command exit.

Table 67. C interface to command exit IHSXTJAV

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
<th>Type (Size)</th>
</tr>
</thead>
<tbody>
<tr>
<td>client_handle</td>
<td>Set this field to the same value that is passed to you in the EGVE_PARAMETER32 structure when your command exit was invoked.</td>
<td>ULONG (4 bytes)</td>
</tr>
<tr>
<td>client_ip_addr</td>
<td>Set this field to the same value as that passed to you in the EGVE_PARAMETERS32 structure when your command exit was invoked.</td>
<td>SZ (9 bytes)</td>
</tr>
<tr>
<td>command_string</td>
<td>Set the first blank delimited token to be the Java class name to be launched on the topology console.</td>
<td>PSZ (4 bytes)</td>
</tr>
<tr>
<td>server_ip_addr</td>
<td>Set this field to the same value as that passed to you in the EGVE_PARAMETERS32 structure when your command exit was invoked.</td>
<td>SZ (9 bytes)</td>
</tr>
</tbody>
</table>

Note: For data type definitions, see Table 60 on page 120.

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 IhsSend to send command responses to the topology console.

Table 68 illustrates the C interface to the IHSXTJCR command exit.

Table 68. C interface to command exit IHSXTJCR

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
<th>Type (Size)</th>
</tr>
</thead>
<tbody>
<tr>
<td>client_handle</td>
<td>Set this field to the same value that is passed to you in the EGVE_PARAMETER32 structure when your exit was invoked.</td>
<td>ULONG (4 bytes)</td>
</tr>
<tr>
<td>client_ip_addr</td>
<td>Set this field to the same value as that passed to you in the EGVE_PARAMETERS32 structure when your exit was invoked.</td>
<td>SZ (9 bytes)</td>
</tr>
</tbody>
</table>
Table 68. C interface to command exit IHSXTJCR (continued)

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
<th>Type (Size)</th>
</tr>
</thead>
<tbody>
<tr>
<td>correlation_id</td>
<td>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_rsp 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.</td>
<td>struct EGVE_CORR_ID (4 bytes)</td>
</tr>
<tr>
<td>server_ip_addr</td>
<td>Set this field to the same value as that passed to you in the EGVE_PARAMETERS32 structure when your command exit was invoked.</td>
<td>SZ (9 bytes)</td>
</tr>
<tr>
<td>user_ctrl_data</td>
<td>Pointer to structure EGVE_CMD_RSP_CTRL.</td>
<td>PBYTE (4 bytes)</td>
</tr>
<tr>
<td>user_ctrl_data_length</td>
<td>Size of structure EGVE_CMD_RSP_CTRL.</td>
<td>USHORT (2 bytes)</td>
</tr>
<tr>
<td>user_data</td>
<td>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.</td>
<td>PBYTE (4 bytes)</td>
</tr>
<tr>
<td>user_data_length</td>
<td>Size of the serialized structures pointed to by user_data.</td>
<td>USHORT (2 bytes)</td>
</tr>
</tbody>
</table>

Note: For data type definitions, see Table 60 on page 120.

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 10. Using the NetView Management Console Command Profile Editor" on page 92 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 Windows: %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.
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
```

```
tserver config
  -d
  -h xx
  -f xx.xx.xx.xx
```

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.
cpebatch

Format

cpebatch

\text{cpebatch} \text{ filename} \quad \text{-O} \quad \text{-I} \quad \text{-U: loginID} \quad \text{-P: password} \quad \text{-L1: logfile} \quad \text{-V} \quad \text{-M} \quad \text{-G}

Purpose

Starts the command profile editor batch utility. For additional information about the command profile editor, see "Chapter 10. Using the NetView Management Console Command Profile Editor" on page 93.

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

If the user ID is not present in the command line, you are prompted to provide this information. 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. If a password is not in the command line, you are prompted to provide this information. 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. 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 Windows: %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:

<table>
<thead>
<tr>
<th>RC</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Successful. The utility completed successfully. Unless -V was specified in the cpebatch command, the database was updated.</td>
</tr>
<tr>
<td>4</td>
<td>Warning. The database was updated (unless -V specified) and warning messages were logged.</td>
</tr>
<tr>
<td>8</td>
<td>Error. The database was not updated and error messages were logged.</td>
</tr>
<tr>
<td>12</td>
<td>Severe Error. The database was not updated and a severe error terminated the program immediately.</td>
</tr>
</tbody>
</table>
dbtransfer

Format

dbtransfer
tserver 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
    tserver hostcmd "command_string"
    -h NetView_hostname
    -d NetView_domain_name
    -u NetView_operator_id
    -p NetView_password
```  

**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 NetView_operator_id**
  - 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 NetView_password**
  - 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 80 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 Windows: `%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.

See “Defining the NetView for z/OS User ID and Password on the Topology Server” on page 17 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 Windows: `%BINDIR%\TDS\server\sample`
- For UNIX: `$BINDIR/TDS/server/sample`
hostcmdoper

Format

```
hostcmdoper
```

```
tserver hostcmdoper [ -u NetView_operator_ID ] [ -p NetView_password ]
```

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 ihsshstc.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 ihsshstc.cfg file. This operand cannot be specified with any other operand.
- `-d` Deletes the ihsshstc.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

    tserver ihszfmt logFileName [ -b ] >outputFileName

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 Windows: %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

```
-helpe 
-? 
-h
```

Specifies a help menu to be displayed describing all the options you can specify with the `ihszset` command.
ihszsett

Format

ihszsett

 tserver ihszsett

Purpose

Starts the graphical user interface, which enables you to set the trace options for the topology server.

Note: This command is not available on the Linux for zSeries platform.
service

Format

    service

            service account_name password

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 ‘DomainName\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_KERNEL_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_KERNEL_DRIVER or SERVICE_FILE_SYSTEM_DRIVER, this
parameter is ignored.
**start**

**Format**

```
start
```

```
tserver start -d -y -b -i -p portnumber
```

**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.

*Note:* The server on Linux for zSeries runs only in IP mode.

- `-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.

*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

    tserver stop -f

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

tserver tcpipkey

-\(d\)
-\(o\) \[xxsendxx | xxrecvxxx\]
-\(v\) \[xxsendxx | xxrecvxxx\]
-? 

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.

xxrecvxx

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.

xxrecvxx

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 Windows: %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 in this appendix. On the Windows platform, the topology server commands are a combination of command files and shell scripts. The tserver prefix ensures that the appropriate environment is set up and then starts the requested command.
utility

Format

```
utility

>>> tserver utility
   -b [on|off]
   -c
   -d
   -f
   -m [username] ["the_message"]
   -p
   -s
```

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 Windows: `%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 Windows: `%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
```

```
  -user <name>
  -password <password>
  -host <host machine>
  -restore
  -admin
  -trace
  -rascomp <value>
  -rastype <value>
  -dump <value>
  -perform
  -key nmc
  -b
  -locale <locale>
  -demo
  -local
  -f
  -c
  -saveViewsLocally
  -noPlugin
  -http
  -debug
  -?
```

**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`:

- **NT** Windows NT and Windows 2000
- **.sh** UNIX

**Parameters**

**path**

Specifies the top level of the topology console installation path:

- For Windows: `%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 81](#) for more information.

- **-password <password>**
  Specifies the topology console sign on password. See [“Using the Topology Console Sign On Window” on page 81](#) 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 81 for more information.

-restore
    Specifies the topology console sign on restore console attribute. See "Using the Topology Console Sign On Window" on page 81 for more information.

-admin
    Specifies the topology console sign on administrator attribute. See "Using the Topology Console Sign On Window" on page 81 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 81 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.
-http
Forces the starting of the Web server regardless of the current console property
settings. This option enables another application to set up the NetView
management console environment before sending locate-resource requests.

-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.

TCONSOLE_JAVAOPTS
Command line argument(s) for Java program. This option is for Tivoli
Customer Support only.
tappxx

Format

tappxx

Path—class_name—args

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:

- NT: Windows NT and Windows 2000
- .sh: UNIX

Parameters

- path: Specifies the top level of the topology console installation path:
  - For Windows: `%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.

- **TCONSOLE_JAVAOPTS**: Command line arg(s) for Java program. This option is for Tivoli Customer Support only.
Appendix C. Launching and Using the NetView Management Console from Other Applications

This appendix describes how to launch and use the NetView management console from other applications using servlets, command-line programs, and scripts provided with the NetView management console. It includes information about the servlets, setting up the NetView management console to use the servlets, and the command-line programs that call the servlets.

Servlets

Two servlets provided with the NetView management console can be used by other applications:

- The check-Web-server servlet, IhsRunning, enables other applications to check whether the NetView management console Web server is running.
- The locate-resource servlet, IhsLocRes, enables other applications to have the NetView management console locate a specific resource. If the resource is found, a view containing the specified resource is opened on the NMC topology console machine to which the request is sent.

The default security access for each servlet can be overridden with a servlet_name.access attribute in the defaultscheme.properties file. The defaultscheme.properties file is located on the NMC topology server in the following directory:

<installRoot>\bin\interp\TDS\Server\db\current\settings.

The values for servlet security access are shown in Table 69, where:

- local Indicates that the servlet can be run only from the local host.
- any Indicates that the servlet can be run from any host.
- disabled Indicates that the servlet cannot be run.

Table 69. Changing servlet security access

<table>
<thead>
<tr>
<th>defaultscheme.properties Attribute</th>
<th>Default Value</th>
<th>Other Valid Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>com.tivoli.ihs.servlet.IhsLocRes.access</td>
<td>local</td>
<td>any, disabled</td>
</tr>
<tr>
<td>com.tivoli.ihs.servlet.IhsRunning.access</td>
<td>local</td>
<td>any, disabled</td>
</tr>
</tbody>
</table>

Setup

For other applications to use the servlets to launch the NetView management console:

1. The NetView management console must be started.
2. The Web server must be started in one of these ways:
   - Checking the Automatically start web server check box on the Web Server page in the Console Properties notebook.
   - Using the http parameter with tconsolexx to force the starting of the Web server; for more information, see "tconsolexx" on page 158.
   - Clicking on the Start web server now button on the Web Server page in the Console Properties notebook.
Using the Servlets

This section describes the syntax for using the servlets directly, and the programs that provide a front end to using the servlets.
IhsLocRes Servlet

Format

IhsLocRes

get http://local_host:port_number/com.tivoli.ihs.servlet.IhsLocRes?name=res_name... (1)

&display=UNIX_display_ID HTTP/1.0

Notes:

1 The name and display parameters can be in any order.

Purpose

This is an HTTP get request that uses the IhsLocRes servlet to locate a resource.

Parameters

local_host:port_number

Specifies the host name and port number of the NetView management console Web server.

com.tivoli.ihs.servlet.IhsLocRes

Specifies the IhsLocRes servlet.

name=res_name

Specifies the name of the resource to be located.

display=UNIX_display_ID

Specifies the display ID of the NetView management console. This parameter is required for UNIX, and must match the DISPLAY environment variable on the UNIX machine.

HTTP/1.0

If specified, indicates that response codes are to be returned.

Return Codes

If the HTTP/1.0 parameter is used, these response codes can be returned:

200 The request has been accepted.

400 The required servlet name was not specified or an incorrect value was specified.

403 The request was not made from the NetView management console host system or the NetView management console is currently disabled.

409 The display ID specified does not match the display ID of the NetView management console.

500 An internal error occurred.
IhsRunning Servlet

Format

IhsRunning

get http://local_host:port_number/com.tivoli.ihs.servlet.IhsRunning?

display=UNIX_display_ID HTTP/1.0

Purpose

This is an HTTP get request that uses the IhsRunning servlet to check whether or not the NetView management console Web server is running.

Parameters

local_host:port_number
   Specifies the host name and port number of the NetView management console Web server.

com.tivoli.ihs.servlet.IhsRunning
   Specifies the IhsRunning servlet.

display=UNIX_display_ID
   Specifies the display ID of the NetView management console. This parameter is required for UNIX, and must match the DISPLAY environment variable on the UNIX machine.

HTTP/1.0
   If specified, indicates that response codes are to be returned.

Return Codes

If the HTTP/1.0 parameter is used, these response codes can be returned:

200 The NetView management console Web server is running.

403 The request was not made from the NetView management console host system or the NetView management console web server is currently disabled.

409 The display ID specified does not match the display ID of the NetView management console.
locRes Java Class

Format

LocRes

```java
java -Dverbose com.tivoli.ihs.cli.locRes resource_name
```

Purpose
The locRes Java class uses the IhsLocRes servlet to locate a resource.

Parameters

- `-Dverbose`
  Specifies that debugging information, including the return code, is to be printed.

`com.tivoli.ihs.cli.locRes`
  Specifies the locRes Java class, which uses the IhsLocRes servlet.

`resource_name`
  Specifies the name of the resource to be located.

`port_number`
  Specifies the port number of the NetView management console Web server. The default is port 80.

Return Codes

0  The request was accepted.

1  The request was rejected because the Web server is not running, access is denied, or the resource name is not valid.

2  A command line argument is in error.

Usage
This program is started from the command line or can be run using the `tappxx` scripts; for information about `tappxx`, see "[tappxx](#)" on page 161. To use the Java command, you must install your own Java environment.

If you use the `tappxx` script, the `-Dverbose` parameter must be specified with the `TCONSOLE_JAVAOPTS` environment variable, rather than on the command line.
nmcRunning Java Class

Format

nmcRunning

```java
-Dverbose -Dpoll
com.tivoli.ihs.cli.nmcRunning

port_number
```

Purpose

The nmcRunning Java class uses the IhsRunning servlet to check whether the NetView management console Web server is running.

Parameters

- `-Dverbose`
  - Specifies that debugging information, including the return code, is to be printed.

- `-Dpoll`
  - Specifies that the program is to poll 10 times before stopping. A ten-second wait occurs between each poll. If this parameter is not specified, the program polls only once.

- `com.tivoli.ihs.cli.nmcRunning`
  - Specifies the nmcRunning program, which uses the IhsRunning servlet.

- `port_number`
  - Specifies the port number of the NetView management console Web server. The default is port 80.

Return Codes

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>The request was accepted.</td>
</tr>
<tr>
<td>1</td>
<td>The request was rejected because the Web server is not running or access is denied.</td>
</tr>
<tr>
<td>2</td>
<td>A command line argument is in error.</td>
</tr>
</tbody>
</table>

Usage

This program is started from the command line or can be run using the tappxx scripts; for information about tappxx, see "tappxx on page 161. To use the Java command, you must install your own Java environment.

If you use the tappxx script, the –D parameters must be specified with the TCONSOLE_JAVAOPTS environment variable, rather than on the command line.
tlocResxx Script

Format

tlocResxx

---tlocResxx---path---port_number---resource_name---args---

Purpose
This script submits a locate-resource request to the NetView management console
Web server, starting the NMC topology console, if necessary, with the appropriate
tconsolexx script. The possible values for xx are:

NT Windows NT and Windows 2000
.sh UNIX

Parameters

path
Specifies the top level of the topology console installation path:
• For Windows: %BINDIR%\..\generic_unix\TDS\client
• For UNIX: $BINDIR/../generic_unix/TDS/client

port_number
Specifies the port number of the NetView management console Web server.

resource_name
Specifies the name of the resource to be located.

Note: If the % (percent sign) character is included in the string for the resource
name, you will need to include two percent sign (%%) characters as the
first is removed on the Windows platform.

args
Specifies optional tconsolexx start-up arguments. These arguments are used if
the NMC topology console has to be started.

Usage
This script supports the following optional environment variables:

TCONSOLE_BACKDOOR
Java code libraries placed at the front of CLASSPATH. This option is for
Tivoli Customer Support only.

TCONSOLE_CLASSPATH
Java code libraries placed at the end of CLASSPATH. Typically, this is used
to define other Java classes to be accessed by the NMC topology console.

TCONSOLE_JAVAOPTS
Command line arguments for the Java program. This option is for Tivoli
Customer Support only.
Appendix D. Sending Commands to Multiple NetView 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 z/OS 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.

![Multiple Domain Dialog Example](image)

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 Windows: %BINDIR%\TDS\server\db\current\settings
   - For UNIX: $BINDIR/TDS/server/db/current/settings
Appendix E. Auditing and the ihsaudit.xml File

The NetView management console includes auditing capability and enables a log of various actions taken on the NMC topology console and NMC topology server which can be used as an audit trail. This auditing function is based on and operated by the NMC topology server. An XML (Extensible Markup Language) file, ihsaudit.xml, that is located in the \%BINDIR%/TDS/server/log directory contains the auditing information. The ihsaudit.xml file can be edited by an ASCII text editor. This log can be displayed in a Web browser, or other application, which supports XML.

Note: For display from a Web browser, the browser must support either CSS (Cascading Style Sheet) or XSL (Extensible Style sheet Language) to enable the display of the log data from the ihsaudit.xml file.

The following NetView management console functions can be audited:

- Starting and stopping of the NMC topology server
- Using the NetView NETCONV command to start and stop communication between the host and server
- Signing on and off of the NMC topology console
- Signing on and off of the command profile editor (CPE) batch utility
- Starting commands
- Command responses
- Open view requests, including locate resource requests
- Open view responses
- Closing views
- View Customization
- Flag change requests, including note changes and list suspended resources requests
- Send message requests
- Aggregation priority and threshold change requests

To specify which functions are to be audited, edit the server.properties file located in the \%BINDIR%/TDS/server/config directory. For more details on editing this file, see "Auditing Functions and the Server.Properties File" on page 19. The size of the ihsaudit.xml file can also be set. For more information about this function in the server.properties file, see "Customizing the Size of the Audit Log" on page 22.

Elements in the ihsaudit.xml file

The audit entry elements in the ihsaudit.xml file are defined in a data type definition (DTD) file, ihsaudit.dtd, which is shipped with the NMC topology server as a sample. The ihsaudit.dtd file is located in one of the following directories:

- For Windows: %BINDIR%/TDS/server/sample
- For UNIX: $BINDIR/TDS/server/sample

Note: Editing this file is not recommended.
The ihsaudit.dtd file lists the data elements contained in the ihsaudit.xml file. A description of each of these elements follows:

**action** ELEMENT

Format

```xml
<!ELEMENT action
    (#CDATA) >
```

Description

General purpose, further qualifies the action that occurred for this audit entry. See "Audit Entry Actions" on page 188 for the possible actions for each audit entry.

Parents

Aggregation, command, cpe, flag, netconv, nmcConsole, nmcServer, sendMsg, viewCust, viewNav

Children

None

**admin** ELEMENT

Format

```xml
<!ELEMENT admin
    (#CDATA) >
```

Description

Determines, during sign on and sign off, if administrator access was requested. Will be either yes or no.

Parents

nmcConsole

Children

None

**aggPri** ELEMENT

Format

```xml
<!ELEMENT aggPri
    (#CDATA) >
```

Description

Aggregation priority was changed for a real resource.

Parents

Aggregation

Children

None

**aggregation** ELEMENT

Format

```xml
<!ELEMENT aggregation
    (action,id,ipAddr,rodmId,((aggPri)|(thresholdDeg,thresholdSevDeg,thresholdUnsat))) >
```

Description

Qualifies an auditEntry to represent a change to the aggregation priority of a real resource or a change to the aggregation thresholds of an aggregate resource.
Parents
auditEntry

Children
One each of (action, id, ipAddr, rodmId). One each of (aggPri) for a real resource and one each of (thresholdDeg, thresholdSevDeg, thresholdUnsat) for an aggregate resource.

auditEntry ELEMENT

Format
<!ELEMENT auditEntry
  (date,
   (aggregation|command|cpe|flag|netconv|nmcConsole|nmcServer|sendmsg|viewCust|viewNav))>

Description
Wrappers one audit log entry.

Parents
auditLog

Children
One date and one of the following: aggregation, command, cpe, flag, netconv, nmcConsole, nmcServer, sendMsg, viewCust, viewNav

auditLog ELEMENT

Format
<!ELEMENT auditLog
  (auditEntry+)>

Description
Wrappers the entire audit log.

Parents
None

Children
One (or more) auditEntry (or entries).

cmd ELEMENT

Format
<!ELEMENT cmd
  (#CDATA)>

Description
Contains the command string to be executed. Before a command string is written to the audit log, if the auditTranslateToBrowseableChars attribute in the server.properties file is set to 1, then a subset of characters found in the command response string that cannot be browsed is translated to a character format that can be browsed. The following are examples:
• & is changed to &amp;
• < is changed to &lt;
• > is changed to &gt;

For more information, see "Customizing Special Characters to be Browsed" on page 23.
cmdResp ELEMENT

Format

<!ELEMENT cmdResp
   (#CDATA) >

Description
Contains the command responses. Before a command response string is written to the audit log, the following server.properties attributes are checked in the following order:

- **auditCarriageReturnReplacement**
  If the auditCarriageReturnReplacement attribute is set to 1, then any carriage return (0x0D), line feed (0x0A), or carriage return immediately followed by a line feed (0x0D0A) found in the command response string is replaced with the characters specified by server.properties attribute auditCarriageReturnReplacementValue.
  For more information, see "Customizing the Replacement Character for Carriage Returns and Line Feeds" on page 23.

- **auditTranslateToBrowseableChars**
  If the auditTranslateToBrowseableChars attribute is set to 1, then a subset of characters found in the command response string that cannot be browsed is translated to a format which can be browsed. The following are examples:
  - & is changed to &amp;
  - < is changed to &lt;
  - > is changed to &gt;

  For more information, see "Customizing Special Characters to be Browsed" on page 23.

- **auditCommandResponseMaxSize**
  If the auditCommandResponseMaxSize attribute is not equal to 0 (zero), then the command response string is truncated to the maximum length specified by the attribute.
  For more information, see "Customizing the Length of Command Responses within the Audit Log" on page 23.

Parents
command

Children
None

command ELEMENT

Format

<!ELEMENT command
   (action,id?,ipAddr?,corrId,res?,rodmId?,menuText?,cmd|cmdResp) >
Description
Qualifies an auditEntry to represent either a command to be started or the command responses.

Parents
auditEntry

Children
One each of the following: action, corrId, (cmd|cmdResp). Optionally, one each of the following: id, ipAddr, res, rodmId, menuText.

corrId ELEMENT

Format
<!ELEMENT corrId
 (#CDATA) >

Description
Contains a numeric correlation ID used to correlate a command that is started with the corresponding command responses.

Parents
command

Children
None

cpe ELEMENT

Format
<!ELEMENT cpe
 (action,id) >

Description
Qualifies an auditEntry to represent a command profile editor (CPE) batch utility sign on.

Parents
auditEntry

Children
One each of action and id.

date ELEMENT

Format
<!ELEMENT date
 (#CDATA) >

Description
Timestamp of the audit log entry in the following format: YYYY-MM-DD HH:MM:SS

Parents
auditEntry

Children
None

extSearch ELEMENT

Format
Description
When a locate resource request is performed, indicates if the extended search option was selected. Will be either yes or no

Parents
viewNav

Children
None

flag ELEMENT

Format

<!--ELEMENT flag
(action,id,ipAddr,((flagMask,flagValue)|(noteMask,note))?) -->

Description
Qualifies an auditEntry to represent one of the following:
- Change to a flag, or flags, of a resource
- Change to a note of a resource
- List suspended resources request

If multiple notes are changed at one time, there is a separate auditEntry for each changed note.

Parents
auditEntry

Children
One each of the following: action, id, ipAddr. Optionally, one each of the following: flagMask, flagValue, noteMask, note.

flagMask ELEMENT

Format

<!--ELEMENT flagMask
(#CDATA) -->

Description
Contains a hexadecimal 4-byte field, with one or more bits turned on. It represents the flag or flags that changed (not the value to which the flag changed). These bit values are defined by the UserStatus field in RODM. An example is 0x40000000, which indicates that the suspended flag is being changed.

Parents
flag

Children
None

flagValue ELEMENT

Format

<!--ELEMENT flagValue
(#CDATA) -->
Description
Contains a hexadecimal 4-byte field, with one or more bits turned on or
off. This represents the value to which the flag or flags are being changed.
The value in the flagMask field represents the flag or flags being set. These
values are defined by the UserStatus field in RODM. An example is
0x40000000, which turns on the suspended flag.

Parents
flag

Children
None

fromHostname ELEMENT

Format
<!ELEMENT fromHostname
 (#CDATA) >

Description
Contains a TCP host name from which a message is being sent.

Parents
sendMsg

Children
None

fromId ELEMENT

Format
<!ELEMENT fromId
 (#CDATA) >

Description
Contains the user name (a NetView for z/OS logon id) from which a
message is being sent. This is in the context of the send message function.

Parents
sendMsg

Children
None

fromIpAddr ELEMENT

Format
<!ELEMENT fromIpAddr
 (#CDATA) >

Description
Contains the IP address of the user name (a NetView for z/OS logon id)
from which a message is being sent. This is in the context of the send
message function.

Parents
sendMsg

Children
None
hostname ELEMENT

Format

<!ELEMENT hostname
(#CDATA) >

Description
Contains a TCP host name.

Parents
netconv, nmcConsole

Children
None

id ELEMENT

Format

<!ELEMENT id
(#CDATA) >

Description
Contains a user name (a NetView for z/OS logon id).

Parents
aggregation, command, cpe, flag, nmcConsole, viewCust, viewNav

Children
None

initRes ELEMENT

Format

<!ELEMENT initRes
(#CDATA) >

Description
Contains the initiating resource name; for example, the RODM object ID of
the resource that was selected, to navigate to another view.

Parents
viewNav

Children
None

ipAddr ELEMENT

Format

<!ELEMENT ipAddr
(#CDATA) >

Description
Contains a dotted decimal IP address.

Parents
aggregation, command, flag, netconv, nmcConsole, viewCust, viewNav

Children
None
locateName ELEMENT

Format

<!ELEMENT locateName

(CDATA) >

Description
Contains the name of the resource that was entered on the Locate Resource
dialog. Before a locate name is written to the audit log, if the
auditTranslateToBrowseableChars attribute in the server.properties file is
set to 1, then a subset of characters found in the name that cannot be
browsed is translated to a format that can be browsed. The following are
examples:
• & is changed to &amp;
• < is changed to &lt;
• > is changed to &gt;

For more information, see "Customizing Special Characters to be Browsed" on page 23.

Parents
viewNav

Children
None

lu62name ELEMENT

Format

<!ELEMENT lu62name

(CDATA) >

Description
Contains the LU 6.2 address.

Parents
netconv

Children
None

menuText ELEMENT

Format

<!ELEMENT menuText

(CDATA) >

Description
Contains the menu text of the command being started. Before a menu text
is written to the audit log, if the auditTranslateToBrowseableChars
attribute in the server.properties file is set to 1, then a subset of
characters that cannot be browsed is found in the menu text and translated
into characters that can be browsed. The following are examples:
• & is changed to &amp;
• < is changed to &lt;
• > is changed to &gt;

For more information, see "Customizing Special Characters to be Browsed" on page 23.

Appendix E. Auditing and the ihsaudit.xml File 181
Parents
command

Children
None

msg ELEMENT

Format
<![CDATA]>

Description
Contains the message that was sent, using the send message function. Before a message is written to the audit log, if the auditTranslateToBrowseableChars attribute in the server.properties file is set to 1, then a subset of characters in the message that cannot be browsed is found and translated into characters that can be browsed. The following are examples:
• & is changed to &amp;
• < is changed to &lt;
• > is changed to &gt;

For more information, see "Customizing Special Characters to be Browsed" on page 23.

Parents
sendMsg

Children
None

netconv ELEMENT

Format
<![CDATA]>

Description
Qualifies an auditEntry to represent a change in the NETCONV connection between NetView for z/OS and the NMC topology server (from the perspective of NMC topology server).

Parents
auditEntry

Children
One each of the following: action, nvDomain. Optionally, if the NETCONV connection is with:
• IP, one each of the following: ipAddr, hostname.
• LU 6.2., then one each of lu62name

nmcConsole ELEMENT

Format
<![CDATA]>

Parents
Children
None

NetView Management Console User’s Guide
Description
Qualifies an auditEntry to represent the sign on or sign off of a NetView management console.

Parents
auditEntry

Children
One each of the following: action, id, ipAddr, hostname, admin.

nmcServer ELEMENT

Format
<!ELEMENT nmcServer
(action) >

Description
Qualifies an auditEntry to represent the start or stop of the NMC topology server.

Parents
auditEntry

Children
One of action.

note ELEMENT

Format
<!ELEMENT note (#CDATA) >

Description
Contains the free-form note text that was changed for the flag of a resource. If the note was made null, this element exists with no data in it. The value in the noteMask field represents the flag (bit) to which this note pertains. Before a note is written to the audit log, if the auditTranslateToBrowseableChars attribute in the server.properties file is set to 1, then a subset of characters in the note that cannot be browsed is found and translated into characters that can be browsed. The following are examples:
• & is changed to &amp;
• < is changed to &lt;
• > is changed to &gt;

For more information, see "Customizing Special Characters to be Browsed" on page 23.

Parents
flag

Children
None

noteMask ELEMENT

Format
<!ELEMENT noteMask
(#CDATA) >

Description
Contains a hexadecimal 4-byte field, with one bit turned on, that
represents the note that was changed. An example is 0x40000000. This indicates that the note for the suspended flag is changed.

Parents
flag

Children
None

**nvDomain Element**

Format
```
<!ELEMENT nvDomain
   (#CDATA) >
```

Description
Contains a NetView for z/OS domain id.

Parents
netconv

Children
None

**res ELEMENT**

Format
```
<!ELEMENT res
   (#CDATA) >
```

Description
Contains the display name of a resource at the server.

Parents
command

Children
None

**rodmId ELEMENT**

Format
```
<!ELEMENT rodmId
   (#CDATA) >
```

Description
Contains the hexadecimal RODM object id, for example, 0000000E00000466.

Parents
aggregation, command, flag

Children
None

**sendMsg ELEMENT**

Format
```
<!ELEMENT sendMsg
   (action,fromId,fromIpAddr,fromHostname,(toId,toIpAddr,toHostname)?,msg) >
```

Description
Qualifies an auditEntry to represent a message that was sent, using the send message function.
thresholdDeg ELEMENT
Format
<!ELEMENT thresholdDeg (#CDATA) >
Description
The degraded threshold was changed for an aggregate resource.
Parents
aggregation
Children
None

thresholdSevDeg ELEMENT
Format
<!ELEMENT thresholdSevDeg (#CDATA) >
Description
The severely degraded threshold was changed for an aggregate resource.
Parents
aggregation
Children
None

thresholdUnsat ELEMENT
Format
<!ELEMENT thresholdUnsat (#CDATA) >
Description
The unsatisfactory threshold was changed for an aggregate resource.
Parents
aggregation
Children
None

toHostname ELEMENT
Format
<!ELEMENT toHostname (#CDATA) >
Description
Contains a TCP host name to which a message is being sent.
Parents
  sendMsg
Children
  None

toId ELEMENT

Format
  <!ELEMENT toId
     (#CDATA) >

Description
  Contains the user name (a NetView for z/OS logon id) to which a message
  is sent. This is within the context of the send message function.

Parents
  sendMsg
Children
  None
toIpAddr ELEMENT

Format
  <!ELEMENT toIpAddr
     (#CDATA) >

Description
  Contains the IP address of the user name (a NetView for z/OS logon id)
  from which a message is being sent. This is in the context of the send
  message function.

Parents
  sendMsg
Children
  None
viewCust ELEMENT

Format
  <!ELEMENT viewCust
     (action,id,ipAddr,viewName) >

Description
  Qualifies an auditEntry to represent a view that was customized and saved
  at the NMC topology server. The view customization was saved, removed,
  or deleted.

Parents
  auditEntry
Children
  One each of the following: action, id, ipAddr, viewName.

viewName ELEMENT

Format
  <!ELEMENT viewName
     (#CDATA) >
Description
Contains a name of a view. Before a view name is written to the audit log, if the auditTranslateToBrowseableChars attribute in the server.properties file is set to 1, then a subset of characters in the view name that cannot be browsed is found and translated to a format that can be browsed. The following are examples:
• & is changed to &amp;
• < is changed to &lt;
• > is changed to &gt;
For more information, see "Customizing Special Characters to be Browsed" on page 23.

Parents
viewCust, viewNav

Children
None

viewNav ELEMENT

Format
<!ELEMENT viewNav
  (action,id,ipAddr,type,((viewName)|(initRes)|(locateName,extSearch))) >

Description
Qualifies an auditEntry to represent view navigation. Examples include: opening a network or exception view, double-clicking a resource for more detail, configuration parents, locating a resource, closing a view.

Parents
auditEntry

Children
One each of action, id, ipAddr, viewType. One of the following:
• One each of locateName and extSearch
• One of initRes
• One of viewName

viewType ELEMENT

Format
<!ELEMENT viewType
  (#CDATA) >

Description
Contains the type of view.

Parents
viewCust, viewNav

Children
None
Audit Entry Actions

The actions for the elements are explained in the following sections. All of the audit entries are children of the auditEntry element. Though action elements are strings, they are considered an extension of the auditEntry element and are, therefore, not translated. If the attribute (as shown in the following tables) in the server.properties file is equal to 1 (one), then the corresponding action is written to the audit log.

aggregation ELEMENT

The action element data for an aggregation audit entry is described in Table 70.

Table 70. Description of the aggregation action element

<table>
<thead>
<tr>
<th>Action element data</th>
<th>Other elements</th>
<th>Description</th>
<th>Attribute</th>
</tr>
</thead>
<tbody>
<tr>
<td>priority</td>
<td>id</td>
<td>User name</td>
<td>Aggregation priority is updated for this resource.</td>
</tr>
<tr>
<td></td>
<td>ipAddr</td>
<td>IP address of console</td>
<td></td>
</tr>
<tr>
<td></td>
<td>rodmId</td>
<td>RODM object ID</td>
<td></td>
</tr>
<tr>
<td></td>
<td>aggPri</td>
<td>Aggregation priority value</td>
<td></td>
</tr>
<tr>
<td>threshold</td>
<td>id</td>
<td>User name</td>
<td>At least one of the aggregation threshold fields is updated for this resource.</td>
</tr>
<tr>
<td></td>
<td>ipAddr</td>
<td>IP address of console</td>
<td></td>
</tr>
<tr>
<td></td>
<td>rodmId</td>
<td>RODM object ID</td>
<td></td>
</tr>
<tr>
<td></td>
<td>thresholdDeg</td>
<td>Degraded threshold value</td>
<td></td>
</tr>
<tr>
<td></td>
<td>thresholdSevDeg</td>
<td>Severely degraded threshold value</td>
<td></td>
</tr>
<tr>
<td></td>
<td>thresholdUnsat</td>
<td>Unsatisfactory threshold value</td>
<td></td>
</tr>
</tbody>
</table>

command ELEMENT

The action element data for a command audit entry is described in Table 71 on page 189.
**Table 71. Description of the command action element**

<table>
<thead>
<tr>
<th>Action element data</th>
<th>Other elements</th>
<th>Description</th>
<th>Attribute</th>
</tr>
</thead>
<tbody>
<tr>
<td>execution</td>
<td>id</td>
<td>User name</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ipAddr</td>
<td>IP address of console, if not a tserver hostcmd</td>
<td></td>
</tr>
<tr>
<td></td>
<td>corrId</td>
<td>Correlation ID to correlate commands to their responses</td>
<td></td>
</tr>
<tr>
<td></td>
<td>res</td>
<td>Resource name if command is resource dependent</td>
<td></td>
</tr>
<tr>
<td></td>
<td>rodmId</td>
<td>RODM object ID if command is resource dependent</td>
<td></td>
</tr>
<tr>
<td></td>
<td>menuText</td>
<td>Menu text of command being executed</td>
<td></td>
</tr>
<tr>
<td></td>
<td>cmd</td>
<td>Command string of the command being executed</td>
<td></td>
</tr>
<tr>
<td>response</td>
<td>id</td>
<td>User name, if not a tserver hostcmd response</td>
<td>auditCommandResponse</td>
</tr>
<tr>
<td></td>
<td>ipAddr</td>
<td>IP address of console, if not a tserver hostcmd response</td>
<td></td>
</tr>
<tr>
<td></td>
<td>corrId</td>
<td>Correlation ID to correlate commands to their responses</td>
<td></td>
</tr>
<tr>
<td></td>
<td>cmdResp</td>
<td>Command response string</td>
<td></td>
</tr>
</tbody>
</table>

A command is being initiated. Commands that are executed in NetView for z/OS are also logged at the NetView program; the auditing of commands and command responses, at the NetView management console server, has no effect on the netlog of NetView for z/OS.

The command response string. There might be one or more audit log entries for command responses, depending on how the target of the command provides chunks of the response. Commands that are started in NetView for z/OS are also logged at the NetView program; the auditing of commands and command responses at the NetView management console server has no effect on netlog of NetView for z/OS.

---

**cpe ELEMENT**

The action element data for a cpe audit entry is described in Table 72.

**Table 72. Description of the cpe action element**

<table>
<thead>
<tr>
<th>Action element data</th>
<th>Other elements</th>
<th>Description</th>
<th>Attribute</th>
</tr>
</thead>
<tbody>
<tr>
<td>signon</td>
<td>id</td>
<td>User name</td>
<td>auditCPESignonSignoff</td>
</tr>
<tr>
<td>signonFailed</td>
<td>id</td>
<td>User name</td>
<td>auditCPESignonSignoff</td>
</tr>
<tr>
<td>signoff</td>
<td>id</td>
<td>User name</td>
<td>auditCPESignonSignoff</td>
</tr>
</tbody>
</table>

---

**flag ELEMENT**

The action element data for a flag audit entry is described in Table 73 on page 190.
Table 73. Description of the flag action element.

<table>
<thead>
<tr>
<th>Action element data</th>
<th>Other elements</th>
<th>Description</th>
<th>Attribute</th>
</tr>
</thead>
<tbody>
<tr>
<td>changeFlag</td>
<td>id</td>
<td>User name</td>
<td>auditFlagChange</td>
</tr>
<tr>
<td></td>
<td>ipAddr</td>
<td>IP address of console</td>
<td></td>
</tr>
<tr>
<td></td>
<td>rodmId</td>
<td>RODM object ID</td>
<td></td>
</tr>
<tr>
<td>flagMask</td>
<td>0xFFFFFFFF value with bits on to represent the flags being changed</td>
<td></td>
<td></td>
</tr>
<tr>
<td>flagValue</td>
<td>0xFFFFFFFF value with value of the bits specified by flagMask</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>A flag (UserStatus field in RODM) was set on a resource. Multiple flags can be set on the resource; all changes are represented in this audit log entry.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>changeNote</td>
<td>id</td>
<td>User name</td>
<td>auditFlagChange</td>
</tr>
<tr>
<td></td>
<td>ipAddr</td>
<td>IP address of console</td>
<td></td>
</tr>
<tr>
<td></td>
<td>rodmId</td>
<td>RODM object ID</td>
<td></td>
</tr>
<tr>
<td>noteMask</td>
<td>0xFFFFFFFF value with bits on to represent the note being changed</td>
<td></td>
<td></td>
</tr>
<tr>
<td>note</td>
<td>Note was changed on the bit represented by noteMask for this resource.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>A note (UserStatusNote field in RODM) was set on a resource. Multiple notes can be set on the resource; a separate audit log entry is written for each note.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>listSuspended</td>
<td>id</td>
<td>User name</td>
<td>auditFlagChange</td>
</tr>
<tr>
<td></td>
<td>ipAddr</td>
<td>IP address of console</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**netconv ELEMENT**

The action element data for a netconv audit entry is described in Table 74.

Table 74. Description of the netconv action element

<table>
<thead>
<tr>
<th>Action element data</th>
<th>Other elements</th>
<th>Description</th>
<th>Attribute</th>
</tr>
</thead>
<tbody>
<tr>
<td>up</td>
<td>nvDomain</td>
<td>NetView domain ID</td>
<td>auditNetconvUpDown</td>
</tr>
<tr>
<td></td>
<td>ipAddr</td>
<td>NetView IP address if over IP</td>
<td></td>
</tr>
<tr>
<td></td>
<td>hostname</td>
<td>NetView TCP host name if over IP</td>
<td></td>
</tr>
<tr>
<td>lu62name</td>
<td>LU name if over LU 6.2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>NETCONV start</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Table 74. Description of the netconv action element (continued)

<table>
<thead>
<tr>
<th>Action element data</th>
<th>Other elements</th>
<th>Description</th>
<th>Attribute</th>
</tr>
</thead>
<tbody>
<tr>
<td>down</td>
<td>nVDomain</td>
<td>NetView domain ID</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ipAddr</td>
<td>NetView IP address if over IP</td>
<td></td>
</tr>
<tr>
<td></td>
<td>hostname</td>
<td>NetView TCP host name if over IP</td>
<td></td>
</tr>
<tr>
<td></td>
<td>lu62name</td>
<td>LU name if over LU 6.2</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>NETCONV stop</td>
<td>auditNetconvUpDown</td>
</tr>
</tbody>
</table>

### nmcConsole ELEMENT

The action element data for an nmcConsole audit entry is described in [Table 75](#).

### Table 75. Description of the nmcConsole action element

<table>
<thead>
<tr>
<th>Action element data</th>
<th>Other elements</th>
<th>Description</th>
<th>Attribute</th>
</tr>
</thead>
<tbody>
<tr>
<td>signon</td>
<td>id</td>
<td>User name</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ipAddr</td>
<td>IP address of console</td>
<td></td>
</tr>
<tr>
<td></td>
<td>hostname</td>
<td>TCP host name of console</td>
<td></td>
</tr>
<tr>
<td></td>
<td>admin</td>
<td>Either yes or no if administrator access was requested</td>
<td>auditConsoleSignonSignoff</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Successful NMC topology console signon</td>
<td></td>
</tr>
<tr>
<td>signonFailed</td>
<td>id</td>
<td>User name</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ipAddr</td>
<td>IP address of console</td>
<td></td>
</tr>
<tr>
<td></td>
<td>hostname</td>
<td>TCP host name of console</td>
<td></td>
</tr>
<tr>
<td></td>
<td>admin</td>
<td>Either yes or no if administrator access was requested</td>
<td>auditConsoleSignonSignoff</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Unsuccessful NMC topology console signon</td>
<td></td>
</tr>
<tr>
<td>signoff</td>
<td>id</td>
<td>User name</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ipAddr</td>
<td>IP address of console</td>
<td></td>
</tr>
<tr>
<td></td>
<td>hostname</td>
<td>TCP host name of console</td>
<td></td>
</tr>
<tr>
<td></td>
<td>admin</td>
<td>Either yes or no if administrator access was requested</td>
<td>auditConsoleSignonSignoff</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NMC topology console signoff</td>
<td></td>
</tr>
</tbody>
</table>

### nmcServer ELEMENT

The action element data for an nmcServer auditEntry is described in [Table 76 on page 193](#).
### Table 76. Description of the nmcServer action element

<table>
<thead>
<tr>
<th>Action element data</th>
<th>Description</th>
<th>Attribute</th>
</tr>
</thead>
<tbody>
<tr>
<td>start</td>
<td>NMC topology server is started</td>
<td>auditServerStartStop</td>
</tr>
<tr>
<td>initialized</td>
<td>NMC topology server is initialized</td>
<td>auditServerStartStop</td>
</tr>
<tr>
<td>stop</td>
<td>NMC topology server is stopped</td>
<td>auditServerStartStop</td>
</tr>
</tbody>
</table>

### sendMsg ELEMENT

The action element data for a sendMsg audit entry is described in Table 77.

### Table 77. Description of the sendMsg action element

<table>
<thead>
<tr>
<th>Action element data</th>
<th>Other elements</th>
<th>Description</th>
<th>Attribute</th>
</tr>
</thead>
<tbody>
<tr>
<td>broadcast</td>
<td>fromId</td>
<td>User name of the console sending the message</td>
<td>auditSendMessage</td>
</tr>
<tr>
<td></td>
<td></td>
<td>or topology server if message was initiated from</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>the NMC topology server</td>
<td></td>
</tr>
<tr>
<td></td>
<td>fromIpAddr</td>
<td>IP address of the console or server sending the</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>message</td>
<td></td>
</tr>
<tr>
<td></td>
<td>fromHostname</td>
<td>Host name of the console or server sending the</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>message</td>
<td></td>
</tr>
<tr>
<td></td>
<td>msg</td>
<td>Message text that was sent</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>A broadcast message was sent.</td>
<td></td>
</tr>
</tbody>
</table>
### Table 77. Description of the sendMsg action element (continued)

<table>
<thead>
<tr>
<th>Action element data</th>
<th>Other elements</th>
<th>Description</th>
<th>Attribute</th>
</tr>
</thead>
<tbody>
<tr>
<td>specificUser</td>
<td>fromId</td>
<td>User name of the console sending the message or topology server if message was initiated from the NMC topology server</td>
<td>A specific message was sent.</td>
</tr>
<tr>
<td></td>
<td>fromIpAddr</td>
<td>IP address of the console or server sending the message</td>
<td></td>
</tr>
<tr>
<td></td>
<td>fromHostname</td>
<td>Host name of the console or server sending the message</td>
<td></td>
</tr>
<tr>
<td></td>
<td>toId</td>
<td>Targeted console user. Could also be &quot;topology server&quot; if targeted to the NMC topology server.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>toIpAddr</td>
<td>Targeted console IP address</td>
<td></td>
</tr>
<tr>
<td></td>
<td>toHostname</td>
<td>Targeted console host name.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>msg</td>
<td>Message text that was sent</td>
<td></td>
</tr>
</tbody>
</table>

### viewCust ELEMENT

The action element data for a viewCust audit entry is described in Table 78.

### Table 78. Description of the viewCust action element

<table>
<thead>
<tr>
<th>Action element data</th>
<th>Other elements</th>
<th>Description</th>
<th>Attribute</th>
</tr>
</thead>
<tbody>
<tr>
<td>save</td>
<td>id</td>
<td>User name</td>
<td>auditViewCustomizationSaveRemove</td>
</tr>
<tr>
<td></td>
<td>ipAddr</td>
<td>IP address of console</td>
<td></td>
</tr>
<tr>
<td></td>
<td>viewName</td>
<td>View name</td>
<td></td>
</tr>
<tr>
<td>remove</td>
<td>id</td>
<td>User name</td>
<td>auditViewCustomizationSaveRemove</td>
</tr>
<tr>
<td></td>
<td>ipAddr</td>
<td>IP address of console</td>
<td></td>
</tr>
<tr>
<td></td>
<td>viewName</td>
<td>View name</td>
<td></td>
</tr>
<tr>
<td>delete</td>
<td>id</td>
<td>User name</td>
<td>auditViewCustomizationSaveRemove</td>
</tr>
<tr>
<td></td>
<td>ipAddr</td>
<td>IP address of console</td>
<td></td>
</tr>
<tr>
<td></td>
<td>viewName</td>
<td>View name</td>
<td></td>
</tr>
</tbody>
</table>
**viewNav ELEMENT**

The action element data for a viewNav audit entry is described in Table 79.

<table>
<thead>
<tr>
<th>Action element data</th>
<th>Other elements</th>
<th>Description</th>
<th>Attribute</th>
</tr>
</thead>
<tbody>
<tr>
<td>openRequest</td>
<td>id</td>
<td>User name</td>
<td>auditViewOpenRequest</td>
</tr>
<tr>
<td></td>
<td>ipAddr</td>
<td>IP address of console</td>
<td></td>
</tr>
<tr>
<td></td>
<td>viewType</td>
<td>Type of view requested, (for example: locateResource, moreDetail, configParents, configChildren, configLogPhy, configLog, ConfigPhy, configBackbone, fastpath, network, exception, configPeers)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>initRes, viewName or locateName</td>
<td>For network and exception views, the audit entry contains viewName which is the name of the view. For Locate Resource, the audit entry contains locateName, which is the resource name as entered by the operator, for example, RALV4. For the more detail, configuration or fastpath views, the audit entry contains initRes which is the RODM id of the initiating resource name.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>extSearch</td>
<td>For locate resource requests, indicates if extended search was requested. Valid values are yes or no.</td>
<td></td>
</tr>
<tr>
<td>Action element</td>
<td>Other elements</td>
<td>Description</td>
<td>Attribute</td>
</tr>
<tr>
<td>----------------</td>
<td>---------------</td>
<td>-------------</td>
<td>-----------</td>
</tr>
<tr>
<td>openResponse</td>
<td>id</td>
<td>User name</td>
<td>auditViewOpenResponse</td>
</tr>
<tr>
<td></td>
<td>ipAddr</td>
<td>IP address of console</td>
<td></td>
</tr>
<tr>
<td></td>
<td>viewType</td>
<td>Type of view request for example: moreDetail, configParents, configChildren, configPeers, configLogPhy, configLog, configPhy, configBackbone, network, exception, fastpath.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>viewName</td>
<td>View name</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>A RODM-based view was opened.</td>
<td></td>
</tr>
<tr>
<td>open</td>
<td>id</td>
<td>User name</td>
<td>auditViewOpenRequest or auditViewOpenResponse</td>
</tr>
<tr>
<td></td>
<td>ipAddr</td>
<td>IP address of console</td>
<td></td>
</tr>
<tr>
<td></td>
<td>viewType</td>
<td>Type of view request, for example server.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>viewName</td>
<td>View name</td>
<td></td>
</tr>
<tr>
<td>close</td>
<td>id</td>
<td>User name</td>
<td>auditViewClose</td>
</tr>
<tr>
<td></td>
<td>ipAddr</td>
<td>IP address of console</td>
<td></td>
</tr>
<tr>
<td></td>
<td>viewType</td>
<td>Type of view, for example: moreDetail, configParents, configChildren, configPeers, configLogPhy, configLog, configPhy, configBackbone, network, exception, fastpath.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>viewName</td>
<td>View name</td>
<td>A view was closed.</td>
</tr>
</tbody>
</table>
Appendix F. 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 topology 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 topology 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 topology console in demo mode (not requiring a NMC topology 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 Windows: %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 topology 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 topology 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 37 for more detailed information.

The lib directory is handled differently than the others. Files installed by the Console, and named in the contents.properties 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 157 for more information about command line arguments to suppress this downloading process.
Appendix G. 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 200] for details.

2. Convert the NGMF command tree facility definition file (.CDF).
   
   See ["Converting the NGMF Command Tree Facility Definition File" on page 201] 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 204] 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 Windows: %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 80].

### Table 80. Command tree facility definition files

<table>
<thead>
<tr>
<th>Feature</th>
<th>Command Tree Facility Definition File</th>
<th>Converted NGMF Response File</th>
</tr>
</thead>
<tbody>
<tr>
<td>NetFinity</td>
<td>FLCH0001.CDF</td>
<td>FLC001HR.RSP</td>
</tr>
<tr>
<td>IP</td>
<td>FLCI0001.CDF</td>
<td>FLC001IR.RSP</td>
</tr>
<tr>
<td>LNM</td>
<td>FLCL0001.CDF</td>
<td>FLC001LR.RSP</td>
</tr>
<tr>
<td>Open</td>
<td>FLCO0001.CDF</td>
<td>FLC001OR.RSP</td>
</tr>
<tr>
<td>TMR</td>
<td>FLCT0001.CDF</td>
<td>FLC001TR.RSP</td>
</tr>
</tbody>
</table>

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 80 on page 199.

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
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 matches a name in the side tabs in the conversion utility dialog where the command indicators are entered.
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 Windows: %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 157 for more information about the `tappxx` command.

**Note:** To download all of the support files necessary for `tappxx` to be available (such as the online help files) from the NMC topology server, sign on using the main NMC topology console at least one time.

The window shown in Figure 23 is displayed.

![Figure 23. IP Resource Specific Commands - Resource Manager Names Dialog](image)

3. In the uppermost text field, type the names of all resource managers that use the commands and command sets. In the Help HTML File section of the dialog, provide the following information:

- In the File Name or Prefix text field, type the file name or prefix of the HTML help file for the command tree facility commands.
- Select one of the following in the Help HTML File section:
  - If the help files should be contained in one flat file, click **Single help file with anchors** and specify the `CommandFileName.HTML` file.
  - If a different help file should be created for each command, click **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 24 on page 203 is displayed.
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 after all indicators are entered. The screen shown in Figure 25 is displayed.

7. Click Yes to verify the command indicators or to change help files. Click No 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 26 on page 204 for each command belonging to a command set and that command set name. Verify the values and make appropriate changes. After you verify the command indicators, click OK to create the response file.
If, at any time during this process, you are satisfied with the verification process or if you choose not to verify the rest of the commands or command sets, you can select Stop verification. This action creates the response file.

**Combining the Output Files**

To combine the two output files created 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
```
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 141.
Index

Special Characters
%hb_data2% 84
%hb_data4% substitution variable 109
%hb_hostname% substitution variable 109
%hb_primary% substitution variable 109
%hb_secondary% substitution variable 109
%hb_sub_origin% substitution variable 109
%hb_sub_source% substitution variable 109
%ipaddress% substitution variable 109
%label% substitution variable 109
%monitor% substitution variable 109
%tme_oid% substitution variable 109
-G option, cpebatch command 142
-I option, cpebatch command 141
-M option, cpebatch command 142
-O option, cpebatch command 141
-P option, cpebatch command 141
-U option, cpebatch command 141
-V option, cpebatch command 142

A
accessibility information xi
adding NMC help 26
adding topology console backgrounds 26
advanced customization, topology console 28
aggregate resource determination 134
aggregate resources 4
animation icon 84
Automatic console file updates 197

B
batch utility
command profile editor 95 starting 96
response file 96
response file defined 95
return codes 142
sample response file 95 starting 96
books
feedback x
online x
ordering x
building 32-bit C programs 112
business tree 84

C
C programming language
building 32-bit programs, command exits 112
command exit functions 123
calendar exits 123
command exits
IhsiInit 123
IhsiInitialize 124
IhsiSend 124
IhsiTerminate 125
IhsiWait 125
return codes 126
installing command exits 113
parameter block (egve_parameters32 structure) 114
writing exits 112
command block
command set keywords 101
operator keywords 103
page keywords 98
profile keywords 103
command block, response file 97
command exit functions, C 123
command exits
available programming languages 111
building 32-bit C programs 112
C, writing exits 113
command profiles 105
description 105
determination, aggregate 134
determination, real 134
determination, resource manager 134
determining information 134
IHSODNATV 129
IHSXTHCE 130
IHSXTJAM 108
IHSXTJAV 108, 132
IHSXTJCR 133
invoking remotely, IhsiSend 124
overview 105
registering, IhsiInit 123
registering, IhsiInitialize 124
return codes 126
sample flows
NetView for z/OS commands 111
substitution variables 108
topology server 105
C 112
flow scenarios 111
unregistering, IhsiTerminate 125
using IHSODGENE 106
using IHSODNATV 107
using IHSXTHCE 107
waiting, IhsiWait 125
writing, C 112
command indicators 93
command profile editor
batch utility
response file defined 95
command profile editor (continued)
batch utility (continued)
return codes 142
command exits 93
command sets 93
main window description 93
operators 94
overview 93
profiles 94
response file 96
sample response file 95
command profiles 105
command set keywords 101
command sets 93
converting 199
command tree facility definition file (.CDF)
converting 201
commands 93
config 140
cpebatch 96, 141
dbtransfer 143
getpd 144
hostcmd 145
hostcmdoper 147
ihsfntm 148
ihssznt 149
ihsszett 150
locRes 167
nmRunning 168
service 151
start 152
stop 153
tapp 161
tconsole 158
tcpipkey 154
tlocRes 169
topology console 157
topology server 139
tserver 155
utility 156
commands notebook 107
Commands notebook 107
config command 140
configuration backbone view 11
configuration child view 8
configuration logical and physical view 10
configuration logical view 9
configuration parent view 8
configuration peer view 9
configuration physical view 10
configuration views 8
configuration views, backbone 11
configuration views, child 8
configuration views, logical 9
configuration views, logical and physical 10
configuration views, parent 8
configuration views, peer 9
configuration views, physical 10
Console updates, automatic from server at logon 197
converting, NGMF command tree facility definition file (.CDF) 201
converting, NGMF response file 200
cpebatch 96
cpebatch command 141
-G option 142
-I option 141
-M option 142
-O option 141
-P option 141
-U option 141
-V option 142
response file 96
return codes 142
creating a demo
NMC 43
Customer Support xi
customized views 12
customizing
NMC 19, 25
online help 26
topology console advanced customization 28
topology console backgrounds 26
topology console icons 25
topology server flat file 19
customizing the ihsshst.cfg file 17

daemon 80
databases, topology server
corrupted 89
creating 90
restoring 90
writing server information 89
dbtransfer command 143
defining
Web browser
properties file 71
Web pages 69
details view, description 85
disability information 85
displaying web browser views 13, 27
e-mail contact xi
egve_parameters32 structure 114
exception views 8
exits, command
C 113
determination, aggregate 134
determination, real 134
determination, resource manager 134
determining information 134
IHSDNATV 129
IHSXTHCE 130, 133
IHSXTJAV 132
overview 105
sample flows
NetView for z/OS commands 111
writing 105, 112
flow scenarios 111
feedback about publications xi
filter bar 86
getpd command 144
get help
adding 26
customizing 26
starting 79
hostcmd command 145
hostcmdoper command 147
IHSDGENE 106
IHSDNATV 107
IHSDNATV command exit 129
IhsLocRes servlet 165
IHSMTTME 85
IhsRunning servlet 166
IHSXTHCE 107
IHSXTHCE command exit 130
IHSXTHCE 130
IHSXTHCE 108
IHSXTHCE 108
IHSXTJAV 132
IHSXTHCE 130
IHSXTJAV command exit 133
ihszfmt command 148
ihszet command 149
ihszet command 150
information, accessibility xi
information, disability xi
invoking
IHSDNATV 129
IHSXTHCE 130
IHSXTHCE 132
IHSXTHC 133
IP Commands
issuing from NetView management console 86
Java application
Web launch 69
Java applications 59
Java plug-ins 59
keyboard, shortcut keys xi
keywords
command set 101
operator 103
page, command block 98
profile 103
languages, programming
command exits 111
launching NMC from other applications 163
locate failing resources 12
locRes command 167
LOGON command 82
LOGON statement, PERMIT keyword 82
manager block, response file 97
manuals
feedback x
online x
ordering x
MIB Browser
accessing from NetView management console 87
more detail views 11
NetView
sending commands 171
NetView for z/OS
sample command exit flow 111
NetView management console
operating 79
NetView management console productivity kit 59
NetView Resource Manager
monitoring NetView tasks 87
NetView tasks
views 87
network views 7
NGMF command sets
converting 199
NMC
creating a demo 43
customizing 19, 25
defining the password and ID on the Topology Server 17
functional overview 3
introduction 3
launching from other applications 163
operating 3
views 7
nmcRunning command 168
objects
command profile editor 93
online help 86
online publications x
operating, NetView management console 79
operator keywords 103
operator objects
defined 94
ordering publications xi
views (continued)
  restricted 13
  RODM 7

W
waiting before terminating
  IhsiWait 125
Web browser
  properties file 71
web browser views
  displaying 13, 27
Web launch Java application 69
Web pages
  defining 69
window, topology console
  animation icon 84
  business tree 84
  filter bar 86
  online help 86
  status area 84
  view area 85
  work space 84
work space area 84
writing server information
  databases 89