Automated Operations Network Customization Guide

Version 5 Release 1
Automated Operations Network Customization Guide

Version 5 Release 1
Tivoli NetView for z/OS Automated Operations Network Customization 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, Advanced Peer-to-Peer Networking, AIX, APPN, BookManager, IBMLink, NetView, OS/390, SecureWay, SP, Tivoli Enterprise, TME, VTAM, and z/OS are trademarks or registered trademarks of International Business Machines 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 or other countries.

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 primarily documents information that is NOT intended to be used as Programming Interfaces of Tivoli NetView for z/OS.

This publication also documents intended Programming Interfaces that allow the customer to write programs to obtain the services of Tivoli NetView for z/OS. This information is identified where it occurs, by an introductory statement to a chapter.
Contents

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

Part 1. AON Overview ..................................................... 1

Chapter 1. Introducing AON ............................................... 3
  Centralized Access to Information .................................... 3
  Dynamic Display Facility (DDF) ......................................... 3
  NetView Operator Interface ........................................... 4
  Focal-Point Services ..................................................... 4
Automated Recovery ........................................................ 5
  Automation and Status Logs ........................................... 6
  NetView Log ................................................................... 6
  Automation Notification .................................................. 6
  Automation Table ........................................................... 7
  Automation Operators .................................................... 7
Notification Operators ...................................................... 8
  Thresholds (SNA Only) .................................................... 8
Large-scale Thresholding .................................................. 9
User Exits ....................................................................... 9
  MVS Console .................................................................. 9
Database Maintenance Facility (DBMAINT) ....................... 9
Tailored Routines and Displays ........................................... 10
  Tailoring the EZLCFG01 Control File ................................. 10
  Using the Common Routines Provided with AON ............... 10
Cross-Domain Logon ........................................................ 10

Part 2. Customizing the Dynamic Display Facility (DDF) .......... 13

Chapter 2. Understanding Dynamic Display Facility (DDF) Design ........................................ ..... 15
  Setting up the Dynamic Display Facility for AON ................ 15
  Understanding the Hierarchical Status Display ................... 16
  Defining Dependencies ................................................... 18
<table>
<thead>
<tr>
<th>Page</th>
<th>Section</th>
</tr>
</thead>
<tbody>
<tr>
<td>19</td>
<td>Defining the Priority and Color of the Resources</td>
</tr>
<tr>
<td>21</td>
<td>Updating Status</td>
</tr>
<tr>
<td>22</td>
<td>Dynamic Updates</td>
</tr>
<tr>
<td>22</td>
<td>Problem Resources</td>
</tr>
<tr>
<td>22</td>
<td>Defining Status Panels</td>
</tr>
<tr>
<td>22</td>
<td>Defining Multiple Systems</td>
</tr>
<tr>
<td>23</td>
<td>Implementing DDF</td>
</tr>
<tr>
<td>23</td>
<td>Using Generic Implementation</td>
</tr>
<tr>
<td>25</td>
<td>Specific Implementation of DDF</td>
</tr>
<tr>
<td>26</td>
<td>Defining the Contents of DDF</td>
</tr>
<tr>
<td>27</td>
<td>Starting and Stopping DDF</td>
</tr>
<tr>
<td>28</td>
<td>Loading Panels</td>
</tr>
<tr>
<td>28</td>
<td>Loading Tree Structures</td>
</tr>
<tr>
<td>29</td>
<td>Using the Definition Procedure</td>
</tr>
<tr>
<td>29</td>
<td>Modifying the Control File for DDF</td>
</tr>
<tr>
<td>33</td>
<td>Modifying the EZLTREE Tree Structure</td>
</tr>
<tr>
<td>33</td>
<td>Modifying DDF Panels</td>
</tr>
<tr>
<td>34</td>
<td>Adding AON/TCP and SNA to the Main DDF Panel</td>
</tr>
<tr>
<td>34</td>
<td>Adding TCP to the Main DDF Panel</td>
</tr>
<tr>
<td>34</td>
<td>Adding SNA to the Main DDF Panel</td>
</tr>
<tr>
<td>35</td>
<td>Editing EZLPNLS</td>
</tr>
<tr>
<td>37</td>
<td>Chapter 3. Defining Dynamic Display Facility (DDF) Statements</td>
</tr>
<tr>
<td>38</td>
<td>Defining the Panel Hierarchy (EZLTREE)</td>
</tr>
<tr>
<td>40</td>
<td>Defining Initialization Statements (EZLINIT)</td>
</tr>
<tr>
<td>41</td>
<td>Defining the Screen size (SCREENSZ)</td>
</tr>
<tr>
<td>42</td>
<td>Linking the Chain Detail Records (CHAIN)</td>
</tr>
<tr>
<td>43</td>
<td>Defining the Initial Screen (INITSCRN)</td>
</tr>
<tr>
<td>44</td>
<td>Defining the Number of Operators (MAXOPS)</td>
</tr>
<tr>
<td>45</td>
<td>Propagating Status Upward (PROPUP)</td>
</tr>
<tr>
<td>46</td>
<td>Propagating Status Downward (PROPDOWN)</td>
</tr>
<tr>
<td>47</td>
<td>Defining the Temporary Error Limit (TEMPERR)</td>
</tr>
<tr>
<td>48</td>
<td>Defining the Default Function Key Definitions (PFKnn)</td>
</tr>
<tr>
<td>50</td>
<td>Defining the Detail Function Key for the Detail Display (DPFKnn)</td>
</tr>
<tr>
<td>52</td>
<td>Describing Function Keys on the Detail Panel, Part 1 (DPFKDESC1)</td>
</tr>
<tr>
<td>53</td>
<td>Describing Function-Key Text on the Detail Panel, Part 2 (DPFKDESC2)</td>
</tr>
<tr>
<td>54</td>
<td>Defining Default Colors (DCOLOR)</td>
</tr>
<tr>
<td>55</td>
<td>Defining Default Colors for Status Components (EMPTYCOLOR)</td>
</tr>
<tr>
<td>56</td>
<td>Defining Priority and Color Ranges (PRITBLSZ)</td>
</tr>
<tr>
<td>57</td>
<td>Defining Color and Priority Ranges (PRIORITY)</td>
</tr>
<tr>
<td>57</td>
<td>Defining the Panel Statements (EZLPNLS)</td>
</tr>
<tr>
<td>58</td>
<td>Defining New Panels (PANEL)</td>
</tr>
<tr>
<td>60</td>
<td>Locating the Status Component (STATUSFIELD)</td>
</tr>
<tr>
<td>63</td>
<td>Defining the Text Area (STATUSTEXT)</td>
</tr>
<tr>
<td>64</td>
<td>Defining Text Location (TEXTFIELD)</td>
</tr>
<tr>
<td>65</td>
<td>Defining Displayed Text (TEXTEXT)</td>
</tr>
<tr>
<td>66</td>
<td>Defining Function Keys on the Status Panel (PFKnn)</td>
</tr>
<tr>
<td>68</td>
<td>Defining the End of a Panel (ENDPANEL)</td>
</tr>
<tr>
<td>69</td>
<td>Including Additional Members %INCLUDE</td>
</tr>
<tr>
<td>71</td>
<td>Chapter 4. Implementing Dynamic Display Facility (DDF)</td>
</tr>
<tr>
<td>71</td>
<td>Displaying Network Status on Multiple Panels</td>
</tr>
<tr>
<td>73</td>
<td>Understanding the Multiple Panel Display Function</td>
</tr>
<tr>
<td>73</td>
<td>Defining Multiple Panel Displays</td>
</tr>
<tr>
<td>75</td>
<td>Displaying Network Status on a Single Panel</td>
</tr>
<tr>
<td>77</td>
<td>Understanding How Single-panel Displays Work</td>
</tr>
<tr>
<td>77</td>
<td>Defining Single Panel Displays</td>
</tr>
<tr>
<td>79</td>
<td>Defining a MessageView Display</td>
</tr>
<tr>
<td>81</td>
<td>Implementing DDF in a Focal Point Environment</td>
</tr>
<tr>
<td>81</td>
<td>Viewing a Focal-point Environment</td>
</tr>
</tbody>
</table>
Chapter 5. Issuing Dynamic Display Facility (DDF) Commands

Using the Dynamic Display Facility (DDF) ................................ 98
Adding Status Descriptors (DDFADD) .................................. 99
Clearing DDF (DDFCLEAR) .................................................. 102
Deleting Status Descriptors (DDFDEL) ................................ 104
Loading a Panel Member (DDFPANEL) ............................... 106
Querying Status Descriptors (DDFQRY) .............................. 107
Loading Tree Members (DDFTREE) .................................... 109
Assigning a Problem to an Operator in DDF (MARK) ............ 110
Removing an Operator Assignment in DDF (UNMARK) ........... 111

Part 3. Customizing AON

Chapter 6. Creating Customized Procedures

How Programs Use AON Control File Routines ....................... 116
Performing Program Initialization Processing ....................... 117
Determining If Automation Is Available ................................ 118
Performing Program Unique Logic ..................................... 118
Sample AON Extended Module ........................................... 119

Chapter 7. Using AON Command Processors

Control File Interface Command (EZLCFG) .......................... 124
Log File Interface Command (EZLLOG) .............................. 130
Status File Interface Command (EZLSTS) ............................ 133
Using the SNBU Automation Status File Interface (FKVSNBU) .... 137

Chapter 8. Coding Common Routines

Using the Common Global Variable Command Processor (CGLOBAL) 143
Querying Command Availability (EXIST) ............................ 144
Calculating Time (EZLEATDF) .......................................... 145
Forwarding User Messages (EZLE1UFW) ............................. 146
Checking the Timer (EZLEACKT) ....................................... 147
Recovering Generic Resources (EZLEAGEN) ......................... 148
Getting Resource Information (EZLEAGRN) ......................... 149
Updating the Status File and Logging Messages (EZLEASLN) .. 151
Checking Thresholds (EZLEATHRR) .................................... 156
INFORM Action (EZLECALL) ............................................ 160
Using Active Monitoring and Recovery (EZLECATV) .............. 161
Checking Automation (EZLECAUT) ................................... 163
Processing Generic Failures (EZLEFAIL) ............................ 164
Managing Automation Tables (AUTOCMD/EZLEF002) .............. 167
Setting Panel Message Color (EZLEMCOL) ......................... 170
Formatting Panel Messages (EZLEMSG) .............................. 171
Driving the Inform Policy (EZLEFRM) ............................... 172
Notify Policy List (EZLENTFY) ........................................ 173
Setting the AIP User Status Bit (EZLEAIP) ......................... 174
Routing Commands over Cross-Domain Sessions (EZLERCMD) ... 176
Recovering Resources (EZLRECVC) ................................... 177
Preface

This document describes customization and programming activities used to tailor and extend the automated operations capabilities of the Tivoli® NetView® for z/OS™ V5R1 Automated Operations Network (AON).

AON provides event-driven network automation implemented from a NetView base.

Who Should Read This Document

This document is written for system programmers and lead operations personnel who are responsible for customizing AON. To use this document, you should have experience using NetView and you should understand the requirements of your network.

What This Document Contains

This book includes the following parts, which contain chapters, and appendixes:

- “Part 1. AON Overview” on page 1 provides an overview of AON customization.
- “Part 2. Customizing the Dynamic Display Facility (DDF)” on page 13 describes the dependencies and implementation of DDF.
- “Part 3. Customizing AON” on page 113 explains how to further extend AON automation.
- “Appendix A. VTAM Messages” on page 259 lists the VTAM® messages that AON/SNA uses.
- “Appendix B. Sample AON/SNA SNBU Modem Configurations” on page 265 lists sample modem configurations.

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>IBM®</th>
<th><a href="http://www.ibm.com/">http://www.ibm.com/</a></th>
</tr>
</thead>
<tbody>
<tr>
<td>Tivoli Systems</td>
<td><a href="http://www.tivoli.com/">http://www.tivoli.com/</a></td>
</tr>
<tr>
<td>Tivoli NetView for z/OS</td>
<td><a href="http://www.tivoli.com/nv390">http://www.tivoli.com/nv390</a></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:
Preface

- 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:

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

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:

http://www.ibm.com/shop/publications/order

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

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/

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:

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

The handbook provides information about how to contact Tivoli Customer Support, depending on the severity of your problem, and the following information:
- Registration and eligibility
- Telephone numbers and e-mail addresses, depending on the country you are in
- What information you should gather before contacting support
Note: Additional support for Tivoli NetView for z/OS is available at the NetView for z/OS Web site:

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.

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
Preface

- 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:


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/

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 2. 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**

```
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**

```
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**

```
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 xiv* 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 xv 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 xvi, 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 xiv.

For example, to specify the BOSESS command with the \texttt{sessid} variable, enter:
\verbatim
NCCF BOSESS applid,,sessid
\endverbatim

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 3 for the appearance of syntax elements.

<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>

\[values\]
Abbreviations

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

## Chapter 1. Introducing AON

- Centralized Access to Information .......................................................... 3
- Dynamic Display Facility (DDF) ................................................................. 3
- NetView Operator Interface ........................................................................... 4
- Focal-Point Services ....................................................................................... 4
- Automated Recovery ......................................................................................... 5
  - Automation and Status Logs ........................................................................ 6
  - NetView Log ................................................................................................ 6
  - Automation Notification ................................................................................. 6
  - Automation Table .......................................................................................... 7
  - Automation Operators .................................................................................... 7
  - Notification Operators ................................................................................... 8
  - Thresholds (SNA Only) .................................................................................. 8
  - Large-scale Thresholding .............................................................................. 9
  - User Exits ....................................................................................................... 9
  - MVS Console .................................................................................................. 9
  - Database Maintenance Facility (DBMAINT) .................................................. 9
- Tailored Routines and Displays ..................................................................... 10
  - Tailoring the EZLCFG01 Control File .......................................................... 10
  - Using the Common Routines Provided with AON ....................................... 10
- Cross-Domain Logon ....................................................................................... 10
Chapter 1. Introducing AON

The Automated Operations Network (AON) provides a comprehensive set of programs that can be customized and extended to provide network automation. The following components of AON provide consistent automation across multiple network protocols:

- SNA Automation (AON/SNA)
- TCP/IP Automation (AON/TCP)

AON intercepts alerts and messages that indicate problems with network resources. AON attempts to recover failed resources and monitor resources until they recover. After a resource has recovered, the components of AON keep a record of the resource failures to track recurring network problems.

To automate network functions, AON provides facilities for:
- Online access to centralized information
- Automated problem determination and recovery actions
- User-written routines and user-tailored panels
- Online tutorials and product information through the operator interface.

The following sections describe the programs that enable AON to automate MVS networks.

Centralized Access to Information

AON enables users to gain access to network information from panels and issue commands to multiple systems from a single terminal. This centralized control is provided by:

- Dynamic Display Facility (DDF)
- Focal-point services
- NetView operator interface

Dynamic Display Facility (DDF)

The DDF is the central focus of resource status information. It displays network resources currently being acted on by AON in an exception based hierarchical panel display. As you add new resources to your network, DDF can automatically reflect those resources without requiring you to change AON control file entries. AON also updates message notifications held by the command facility (NCCF).

The DDF enterprise-wide resource display provides real-time, exception-oriented status monitoring. An operator at the focal-point NetView can view the status of multiple networks from the DDF panels. AON dynamically updates DDF panels with resource status information. Multiple programs can update DDF with resource status without concern for the sequence or priorities posted by other programs. Many operators can view DDF and receive updates without having an impact on the network.

AON provides default settings for DDF to represent the status of your network resources. Default DDF settings display only resources that require operator intervention. The mark function enables operators to take responsibility for a failed
resource that requires operator intervention. DDF is updated to indicate who is working on the problem. When failed components recover, DDF automatically removes them from the display.

DDF displays are color-coded to indicate the status of the network components. The color indicates the severity of a failure. If AON detects a problem with a resource, it displays the resource in red, pink, or yellow (default color definitions). Typically, a failure displayed in red is more severe than a failure displayed in yellow. Green (or not displayed) indicates that the resource status is normal.

You can customize DDF settings, including colors. For example, by making a change to the control file you can make DDF display all resources instead of just those that require intervention.

For more information on customizing and using DDF, see “Part 2. Customizing the Dynamic Display Facility (DDF)” on page 13.

NetView Operator Interface
With the NetView operator interface to AON, users can:
• Issue commands
• Receive responses for functions provided by AON and other NetView facilities

Operators can enter commands either in line mode (on any NetView command line) or in full-screen mode. Full-screen mode provides easy-to-use command pop-ups.

Additionally, NetView management console (NMC) operators can access an AON window to display automated resource recovery information and control file information for SNA resources.

Focal-Point Services
You can specify one domain as the focal point in a multidomain network. AON routines forward notification messages from multiple hosts to a single focal-point host. This enables a network operator to receive all the network notifications at a single console.

To support focal points, you define routes, called gateways, from one NetView to another. AON forwards automation notifications, commands, and responses through these gateways.

To implement focal-point services, you define a hierarchical domain structure composed of a focal-point domain and distributed domains. You can also define a backup for the focal-point host, for situations when the primary focal-point host is unavailable. Through focal-point services, you can:
• Enable operators to send commands and receive responses through the gateways. This eliminates the need for personal NetView-NetView task (NNT) sessions for each operator.
• Manage the Resource Access Control Facility (RACF) passwords for gateway automation operators
• Set up and initiate automated operator NNT sessions
• Display the status of gateway automation operators and user NNT sessions
Automated Recovery

This section describes the AON generic recovery processes. AON provides generic automated recovery processes used by the components of AON, which enable you to focus on other areas of operations. AON automation provides the following benefits:

- Fast recovery
- Great network availability
- Automatic responses to predefined messages
- Reminder notification of unavailable resources or resources that are in manual recovery mode
- Limited operator intervention
- AON categorization recovery as critical or non-critical
- More time for the operations staff to work in other areas where automation is not currently available

Although AON can help you automate and manage several different types of networks, the tasks required to automate these networks are very similar in nature. There are different methods of getting resource status in each type of network, but the basic structure of the automation is essentially the same.

AON includes routines that perform similar tasks on different types of networks. This means that the automation process for systems network architecture (SNA), Advanced Peer-to-Peer Networking® (APPN®), or Transmission Control Protocol/Internet Protocol (TCP/IP) resources is the same, although the actual programs that perform the automation steps might be different. For example, if a resource fails in a SNA or TCP/IP network, information about the resource (such as resource type, connectivity, and status) must be gathered before automated recovery can continue for that resource. Each of these network types has a different program that gathers resource information, but they are called and processed in the same manner by the AON automation routine.

The benefits of having common automation routines are:

- Code reliability
- Transfer of skill across networks (because the process is the same for the different types of networks)
- Easier problem determination

AON automates network operations such as resource recovery. This section provides an overview of how AON facilitates recovery, and describes the tools it uses to accomplish automation of network operators, including:

- Automation table
- Automated tasks
- Logging
- Providing generic failure and recovery routines
- Sending messages to the notification operators when further action is required
- Sending records to the automation log for tracking purposes

AON supports network resource recovery by monitoring critical resources and taking automated action based on tailored criteria. Each automation component recovers specific types of resources. For example, SNA automation recovers SNA (VTAM) resources. AON reacts to adverse conditions of network resources and notifies operators of these conditions, when appropriate.
Recovery criteria can be set based on resource type, resource naming convention, explicit resource name, or network-wide settings. You can select a variety of parameters and options to control when and how recovery takes place.

The AON control file contains automation criteria. There are two ways to update the control file, dynamically or statically:

- You can update the control file dynamically through the POLICY command interface or through the AON operator interface. These changes remain in effect until you reload the control file or until the current NetView-started tasks end.
- You can update the control file statically with a system editor, such as the Interactive System Productivity Facility (ISPF). After editing the control file, you can use the POLICY command to reload the control file, reinitialize AON, or recycle your NetView environment.

### Automation and Status Logs

To improve problem determination productivity, AON uses automation logs and status logs to record the automation process and status of the network resources.

The status log tracks:
- Last 10 failures of a resource, with the time stamp
- Current automation status and threshold exceptions
- Last operator notified about a resource or one that took action on a resource

Users can issue the AON DSPSTS command to display status log information. AON provides a facility (DBMAINT) to maintain this VSAM status file.

The AON automation log records all automation activity. When a resource becomes unavailable or when the resource becomes available again, AON writes availability records to the log. These records indicate whether the action was caused by automation, the Help Desk function, or by an operator. You can issue the AON NLOG command to view the automation log.

### NetView Log

AON writes entries to the NetView log for all significant events relative to the control file. This includes operator messages, internal events, and errors within the control file.

### Automation Notification

AON provides you with a notification policy that can be customized to do the following types of operator notification:
- Issue a message.
- Generate an alert or resolution to the hardware monitor.
- Update the DDF.
- Send an event to the Tivoli Enterprise Console (TEC).
- Send an e-mail request.
- Generate a beeper request.

Operators using NMC can request Alert History to view alerts generated by AON. AON also sets the Automation in Progress status so you can see that automation is attempting to recover the failed resource. Failed resources which cannot be recovered appear in the Operator Intervention Network view (OIV).
### Automation Table

NetView enables you to issue commands based on any incoming message or Management Services Unit (MSU). You can specify criteria that must be satisfied before running a program. The criteria can include the presence of specific message text data, resource names, other message attributes, or specific MSU data.

AON provides a predefined message table structure (included in DSITBL01) with each component adding prescribed messages, or MSUs, and criteria. When any of these prescribed messages and conditions occur, the AON generic failure or recovery routines are invoked to take action. Actions are defined in the AON option definition tables. In some cases, the automation table directly drives a command list.

### Automation Operators

The multiple automation operator design enables AON to divide its workload among several, separately defined automated operators by using the concurrent processing capability of NetView. All AON automation operators are started during initialization and must be active at all times.

The automation operators are identified by unique names corresponding to their responsibilities. For example, some of the automation operators supplied with AON include:

- **AIPOPER**
  Sets and resets the AIP (Automation In Progress) operator status bit in RODM. This bit causes a display pattern to be placed on the object in NetView management console. RODM AIP operators issue the commands necessary to update resource objects in RODM views with the AIP operator status. These operators are also used in the management of the OIV processing.

- **ALRTOPER**
  Sends alerts and resolutions to NetView over an LU 6.2 session.

- **BASEOPER**
  Provides backup for other automation operators.

- **DVIPOPER**
  Used for DVIPA polling processes.

- **GATOPER**
  The outbound gateway operator for automation notification forwarding.

- **INFOPER**
  Serializes the updates to the inform log.

- **MSGOPER**
  Formats and issues AON notifications and DDF updates.

- **NETOPER**
  Initiates routines based on the NetView automation table and AON generic failure and recovery routines.

- **NV6KOP**
  Used for Tivoli NetView for UNIX® automation

- **OIVOPER**
  An optional operator task used by the Operation Intervention View (OIV) function. When enabled, automatically deletes resources from the OIV at specified intervals. Only resources with the display status of satisfactory (129) are removed.
TCPOPER
  Used for TCP/IP automation

TRAPOPER
  Used for trap automation processes

WKSTOPER
  Sends and receives commands and responses between AON and a workstation
  with the interface installed.

X25OPER
  Used by X25 automation processes

For more information, refer to the *Tivoli NetView for z/OS Administration Reference*.

**Notification Operators**

Notification operators are operators identified (in NTFYOP control file entries) to receive messages generated by AON. Notifications are necessary to understand and operate the network. They must be routed to the notification operators or the automation operators.

In a distributed network, notification operators are defined at the focal-point domain. Notification operators can also be defined at the distributed domain. If they are defined at both, notifications are forwarded to notification operators at both the focal point and distributed domains.

If the host that creates the automation notification is not the focal-point domain, the message is forwarded to the focal-point domain. In a single NetView environment, the domain is its own focal point.

When the notification arrives or is generated at the focal-point host, this notification is sent to the notification operators and may be “frozen” on the operator’s screen. AON enables you to specify the types of messages that are frozen on the operator’s screen. When notifications are frozen, operators can use the DM command to remove them from the screen.

The notification operators are defined in the control file by operator name, operator description, and message class. For example:

```
NTFYOP OPER1,OPER='OPERATOR 1',CLASS=10
```

The class of the notification is compared to the class of each defined notification operator. Each notification operator with a class matching the class of the automation notification receives the notification. For example, all AON notifications have a class of 90. Therefore, any notification operators defined with CLASS=90 receive all notifications.

**Thresholds (SNA Only)**

AON tracks resource failures and recoveries in a status file. If the number of errors exceeds a defined threshold over a period of time (which is defined in the THRESHOLDS control file entry), AON alerts the notification operator that the resource is experiencing multiple errors and continues recovery attempts. For critical thresholds, AON stops recovery attempts.

AON uses three types of threshold definitions:

**Critical**

AON uses critical thresholds to stop automated recovery. AON deactivates
the resource and notifies the notification operators. Automated recovery is stopped until an operator reactivates the resource.

**Frequent**
AON uses frequent thresholds to indicate recurring errors that should warrant an operator’s attention. When a resource exceeds a frequent threshold, automated recovery continues, but AON sends a message to the notification operators and makes an entry in the automation log.

**Infrequent**
AON uses infrequent thresholds to indicate that a resource is experiencing random or intermittent errors. Automated recovery continues, but AON sends a message to the notification operators and makes an entry in the automation log. When a resource fails and one of these thresholds is exceeded, AON sends a notification to the defined AON notification operator.

**Large-scale Thresholding**
AON uses the LSTHRESH (large scale thresholding) control file entry to count network-wide events of a particular type, and then sets a threshold on the number of times the event can happen over specific period of time.

**User Exits**
AON is designed to fulfill most automated recovery needs. However, an installation may have particular automation requirements that need to be met. AON provides user exits that you can code for specific recovery processing, monitored intervals, threshold checking, and SNA resource information gathering.

You set user exit values in the control file. The user exits are run from certain AON common routines.

For more information on user exits, see "Chapter 10. AON User Exits" on page 213.

**MVS Console**
AON uses the NetView write-to-operator with reply (WTOR) facility to communicate with and solicit responses from the MVS system operator. Likewise, an MVS system operator can run AON routines and commands from the MVS system console by preceding the command with the subsystem designation assigned to NetView.

**Database Maintenance Facility (DBMAINT)**
AON provides a database maintenance facility (DBMAINT) that performs database maintenance for the NetView hardware monitor and session monitor. You can also use DBMAINT to purge records from the AON status file. This database maintenance includes the deletion of records prior to a specified date and the reorganization of the VSAM database.

DBMAINT uses the ENVIRON SETUP entry in the AON control file to correctly perform database maintenance. The ENVIRON SETUP control file entry must match the way the VSAM database was allocated, or errors occur.

You can use the DBMAINT facility from a full-screen operator interface panel or you can schedule it to run on an automation operator station task. The latter can be achieved by building a Timer entry in the AON control file that runs DBMAINT
at a specific time and date. This process automates database and file maintenance. If also enables you to schedule database maintenance during non-peak hours of operation.

**Tailored Routines and Displays**

You can expand the AON functions to meet additional automation needs unique to your network systems. You can do this by:

- Tailoring the EZLCFG01 control file
- Using the common routines provided with AON

**Tailoring the EZLCFG01 Control File**

NetView loads the DSIPARM member EZLCFG01 control file during initialization. This file contains values such as the notification operator IDs, the automation operator IDs, threshold values for resources, monitoring values, and recovery values. In most cases you can change the control file values without restarting NetView, because the new values become effective as soon as the AON control file is reloaded.

The data in the control file is organized by keywords, and therefore is independent of the program language or operating system. You can set values for individual resources, resource types, or system-wide defaults. This reduces the need for generating a set of variables to contain the information needed for each individual resource. You can input your own data in the AON control file to have a common depository for information used by programs on your network.

When you first install AON, system programmers customize control file entries for their network needs.

For more information, see the *Tivoli NetView for z/OS Automation Guide*.

**Using the Common Routines Provided with AON**

AON includes routines that you can use to code network automation extensions. These routines provide easy-to-use generic functions for expanding automation capabilities beyond those supplied and supported by AON. You can use these generic functions to reduce development time when you create procedures or extend those provided.

Common routines perform automated actions such as transferring information and checking the control file. User-written programs can call common routines from the message table, control file, or an extended routine to accomplish a required task.

For more information on common AON routines, see “Chapter 8. Coding Common Routines” on page 141.

**Cross-Domain Logon**

With AON cross-domain logon (CDLOG), you can log on to all or a select group of domains. CDLOG enables you to select which domains you want.

If CDLOG is unable to establish a session with the domain you selected, it displays the message received when attempting to start the session. After trying to set up sessions with other NetView domains, CDLOG indicates the status of all the domains in the domain list.
To use the AON/SNA Help Desk for resources in another domain, you must define the CDLOG entry. The SNA Help Desk determines which domain owns a resource. For the help desk to do this, however, you must be able to log on to all the domains known to your NetView. Automatic logon has two types of logon procedures:

- Automatic mode
- Semiautomatic mode

Automatic mode takes the operator ID and password from the control file and logs on to the selected domains. Semiautomatic mode requires operator action to select domains and provide passwords.

The AON remote gateway (RGWY command) requires CDLOG definitions to start RMTCMD sessions. Several AON functions use the remote gateway sessions.
Part 2. Customizing the Dynamic Display Facility (DDF)

Chapter 2. Understanding Dynamic Display Facility (DDF) Design ........................................ 15
Setting up the Dynamic Display Facility for AON ................................................................. 15
Understanding the Hierarchical Status Display ................................................................. 16
Defining Dependencies ........................................................................................................... 18
Defining the Priority and Color of the Resources ............................................................... 19
Updating Status ..................................................................................................................... 21
  Dynamic Updates ................................................................................................................. 22
  Problem Resources ............................................................................................................. 22
Defining Status Panels ......................................................................................................... 22
Defining Multiple Systems .................................................................................................... 22
Implementing DDF .................................................................................................................. 23
  Using Generic Implementation ............................................................................................ 23
  Specific Implementation of DDF ......................................................................................... 25
Defining the Contents of DDF ............................................................................................... 26
Starting and Stopping DDF ................................................................................................... 27
Loading Panels ...................................................................................................................... 28
Loading Tree Structures ...................................................................................................... 28
Using the Definition Procedure ........................................................................................... 29
Modifying the Control File for DDF ..................................................................................... 33
Modifying the EZLTREE Tree Structure ............................................................................. 33
Modifying DDF Panels .......................................................................................................... 33
Adding AON/TCP and SNA to the Main DDF Panel ............................................................ 34
  Adding TCP to the Main DDF Panel .................................................................................... 34
  Adding SNA to the Main DDF Panel ................................................................................... 34
Editing EZLPNLS ................................................................................................................... 35

Chapter 3. Defining Dynamic Display Facility (DDF) Statements ........................................... 37
Defining the Panel Hierarchy (EZLTREE) ............................................................................. 38
Defining Initialization Statements (EZLINIT) ................................................................. 40
  Defining the Screen size (SCREENSZ) ................................................................................ 41
  Linking the Chain Detail Records (CHAIN) ....................................................................... 42
  Defining the Initial Screen (INITSCRN) ............................................................................ 43
  Defining the Number of Operators (MAXOPS) ............................................................ 44
Propagating Status Upward (PROPUP) ................................................................................ 45
Propagating Status Downward (PROPDOWN) ................................................................. 46
Defining the Temporary Error Limit (TEMPERR) ............................................................. 47
Defining the Default Function Key Definitions (PFKnn) ...................................................... 48
Defining the Detail Function Key for the Detail Display (DPFKnn) ........................................ 50
  Describing Function Keys on the Detail Panel, Part 1 (DPFKDESC1) ................................. 52
  Describing Function-Key Text on the Detail Panel, Part 2 (DPFKDESC2) ............................ 53
  Defining Default Colors (DCOLOR) .................................................................................. 54
  Defining Default Colors for Status Components (EMPTYCOLOR) ................................... 55
  Defining Priority and Color Ranges (PRIORITY) ............................................................. 56
  Defining Color and Priority Ranges (PRIORITY) ............................................................. 57
Defining the Panel Statements (EZLPNLS) .......................................................................... 57
  Defining New Panels (PANEL) .......................................................................................... 58
  Locating the Status Component (STATUSFIELD) ........................................................... 60
  Defining the Text Area (STATUSTEXT) ............................................................................ 63
  Defining Text Location (TEXTFIELD) .............................................................................. 64
  Defining Displayed Text (TEXTTEXT) .............................................................................. 65
  Defining Function Keys on the Status Panel (PFKnn) ...................................................... 66
  Defining the End of a Panel (ENDPANEL) ....................................................................... 68
  Including Additional Members %INCLUDE ..................................................................... 69

Chapter 4. Implementing Dynamic Display Facility (DDF) .................................................... 71
Displaying Network Status on Multiple Panels .................................................................... 71
Chapter 5. Issuing Dynamic Display Facility (DDF) Commands

Using the Dynamic Display Facility (DDF) ........................................... 98
Adding Status Descriptors (DDFADD) .................................................. 99
Clearing DDF (DDFCLEAR) ................................................................. 102
Deleting Status Descriptors (DDFDEL) ................................................ 104
Loading a Panel Member (DDFPANEL) ............................................... 106
Querying Status Descriptors (DDFQRY) .............................................. 107
Loading Tree Members (DDFTREE) ...................................................... 109
Assigning a Problem to an Operator in DDF (MARK) ......................... 110
Removing an Operator Assignment in DDF (UNMARK) ....................... 111
Chapter 2. Understanding Dynamic Display Facility (DDF) Design

The Dynamic Display Facility (DDF) enables you to identify and focus on a specific problem in the network. The DDF functions enable you to alter or extend your DDF installation. This chapter describes the structure and processing of DDF:

- Understanding the hierarchical display of DDF
- Defining dependencies
- Defining the priority and color of the resources
- Updating status
- Defining panels
- Defining multiple systems
- Implementing DDF
- Defining the contents of DDF
- Understanding the flow of DDF
- Starting and stopping DDF
- Loading panels
- Loading tree structures
- Using the definition procedure

Note: Most of the examples used in this chapter are specific to Systems Network Architecture (SNA) resources, but the concepts apply to all AON automation components. To customize DDF to suit your environment, modify one or more members in DSIPARM and DDF panels in CNMPNL1. Sample members and panels are provided. Those samples contain references to domain CNM01. You cannot substitute system symbolics within any of those samples. Instead, you can use the EZLEISP1 utility shipped in CNMCLST. For more information, refer to the Tivoli NetView for z/OS Installation: Configuring Additional Components.

Setting up the Dynamic Display Facility for AON

The Dynamic Display Facility (DDF) enables you to identify and focus on a specific problem in the network. DDF uses colors to represent the status of various network resources. DDF can track the error, warning, action, or informational states of network resources. DDF uses tree structures to implement the hierarchical display of status information. Although DDF can display the status of all network resources, you can tailor DDF to display only resources that have an error.

To facilitate the installation of DDF, the distribution tape provides samples to set up a typical DDF implementation. This implementation reflects DDF updates for a single domain (no focal point implementation).

You can use three types of definitions to set up DDF. Each of these types has its own file or files. The definition types are:
- Entries in the control file
- Tree structures (one file per domain)
- Panel characteristics (one file per panel)

The sample control file and panels are set up to run for domain CNM01. If your domain ID is different, change all occurrences of CNM01 to your domain ID in the following members:
• All members whose name starts with EZL (such as EZLCFG01 and EZLTREE)
• All members whose name starts with FKVPN (for AON/SNA)
• All members whose name starts with FXXPN (for AON/TCP)

Understanding the Hierarchical Status Display

DDF uses colors to represent the status of various network resources. DDF can track the error, warning, action, or informational states of network resources. Although DDF can display the status of all network resources, you can tailor DDF to display only resources that have an error. This helps you focus on network problems rather than the entire network and save central processing unit (CPU) time.

DDF displays the status of resources (active or inactive). Each type of status is displayed in a different color. For example, inactive resources are red and active resources are green. For more information, see “Defining Default Colors for Status Components (EMPTYCOLOR)” on page 55.

You can tailor DDF to display the status of all resources of a particular type. If you use this design, DDF displays each resource type in a different color. For example, DDF might display a Network Control Program (NCP) problem in red and a line problem in pink. See “Displaying Network Status on a Single Panel” on page 73 for instructions about tailoring DDF to display by resource type.

DDF presents a hierarchical status display. Figure 7 on page 17 illustrates a sequence of sample SNA-specific DDF panels, moving from the domain to the resource level.
In Figure 7 you can see:

1. The Data Center Networks panel is the first DDF panel. In Figure 7, SNA, and TCP/IP represent the SNA and TCP/IP resources in the domain, CNM01. To see the types of SNA resources defined to CNM01, move the cursor to SNA and press the down key, F8.

2. All the monitored SNA resource types within network CNM01 are displayed on the CNM01 Network Status panel. To see the names of the physical units defined to CNM01, move the cursor to PUS and press the down key, F8.

3. The next panel shows the names of the physical units that are experiencing problems. DDF shows each monitored resource in the color of its status. For example PU001 is shown in red if the status is INACTIVE, which requires operator intervention for recovery. To see the detail record, move the cursor to PU001 and press the Detail key, F2.

4. The status details record for PU001 is shown on the Detail Status Display. The physical unit PU001 requires operator intervention because a critical threshold has been reached and automation has stopped.

Figure 7. Sample DDF Panel Flow of SNA Resources

Chapter 2. Understanding Dynamic Display Facility (DDF) Design 17
When you select a problem for analysis, press **F2** (the DDF MARK command) to inform others that you are working on the problem. Also, before you resolve a problem, check DDF to determine whether someone is already working on the problem.

AON uses many different programs to update resource status in DDF regardless of the sequence or priorities posted by other programs. As AON automation resolves the problems, DDF removes the resources from the display.

### Defining Dependencies

DDF uses tree structures to implement the hierarchical display of status information. A tree structure starts with the system name as the root node and a level number of 1. The leaves of the tree represent the monitored resources, and the level numbers reflect which resources depend on each other. DDF generic values can also represent the leaves. Figure 8 shows the tree structure provided with the AON/SNA component and includes member EZLTREE in the DSIPARM. This example also shows the order of dependency.

```
/* NETWORK: CNM01 */
1 CNM01
  2 SYSTEM
     3 GATEWAY
     3 GROUPS
      4 CALIF
       5 LA
       5 SANFRAN
       5 SANDIEGO
      4 NEW YORK
      4 ATLANTA
     3 OPID
     4 OPER1
    3 NETWORK
    4 RESOURCE
    4 SNA
     5 SNA
      6 NCP
      6 LINE
      6 LINKSTA
      6 CDRM
      6 CDRSC
      6 PU
      6 LU
      6 SESSION
      6 APPL
      6 ERR
     5 APPN
      6 CP
      6 EN
      5 X25
       6 X25MCH
```

*Figure 8. DDF Tree Structure*

Color in the tree structure is based on the order of dependencies. To consolidate the status of monitored network resources at the root node, specify the DDF parameter PROPAGATE UP. Figure 8 shows how DDF reflects any color changes for LINE on LINE, SNA, NETWORK, SYSTEM, and CNM01. The color of the root node reflects the most important or critical status in a network operations center. If
all the monitored resources are green, or active and not displayed, DDF shows the root node in green on the Data Center Network panel.

**Defining the Priority and Color of the Resources**

If more than one panel of information is available, DDF places $1 \text{ of } n$ field in the upper right corner of the first Detail Status Display. On any series of detail status displays, DDF displays the most critical status color first. If more than one display has the same status color, DDF displays the most recent first.

Figure 9 on page 20 shows three of these panels. This example shows that DDF has fourteen status display panels.
In Figure 9, notice that the pink and turquoise panels precede the green status panel. During DDF initialization, DDF assigns colors to specific priority ranges as defined in DSIPARM member EZLINIT. DDF orders Detail Status Displays by priority and assigns the following priority ranges to these colors:

**Red**  
Priority 100 to 199

**Pink**  
Priority 200 to 299

**Yellow**  
Priority 300 to 399

**Turquoise**  
Priority 400 to 499

**Green**  
Priority 500 to 599
When a new status is added, DDF issues a DDFADD command with the appropriate priority. For example, when the status descriptor with the priority 350 is added DDF displays yellow.

Status descriptors are connected to the status component in ascending order of priority. Therefore, if DDF assigns two status descriptors for PU with priorities of 120 and 150, DDF displays the one with the priority of 120 first. The PU is displayed in red. If a status component has multiple status descriptors with equal priorities, the status descriptors are chained off the status component in order of arrival time. Use the DDFDEL command to delete status descriptors.

### Updating Status

To define the status and types used to set priority and color, place the definitions in the control file. When an automation event occurs, AON logging routines scan the control file for the DDF entry for that status or type. DDF uses the information from the control file and issues a DDFADD request. Table 4 shows the status, color, and priority as provided with the AON base control file entries. The TCP/IP components add more DDF status types.

**Table 4. Examples of DDF Status Defaults (EZLCFGDS)**

<table>
<thead>
<tr>
<th>Resource Status</th>
<th>Color Defaults</th>
<th>Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACT*</td>
<td>GREEN+</td>
<td>550</td>
</tr>
<tr>
<td>CON*</td>
<td>GREEN+</td>
<td>550</td>
</tr>
<tr>
<td>IIN*</td>
<td>RED</td>
<td>160</td>
</tr>
<tr>
<td>INA*</td>
<td>RED</td>
<td>150</td>
</tr>
<tr>
<td>INO*</td>
<td>RED</td>
<td>160</td>
</tr>
<tr>
<td>NEV*</td>
<td>YELLOW</td>
<td>360</td>
</tr>
<tr>
<td>PAC*</td>
<td>PINK</td>
<td>260</td>
</tr>
<tr>
<td>PCT*</td>
<td>PINK</td>
<td>260</td>
</tr>
<tr>
<td>UP</td>
<td>GREEN+</td>
<td>550</td>
</tr>
<tr>
<td>STOP</td>
<td>RED</td>
<td>150</td>
</tr>
<tr>
<td>RES*</td>
<td>TURQUOISE</td>
<td>460</td>
</tr>
<tr>
<td>RCV*</td>
<td>GREEN</td>
<td>550</td>
</tr>
<tr>
<td>REAC*</td>
<td>PINK</td>
<td>270</td>
</tr>
</tbody>
</table>

**Note:**

+ Indicates that the status of resources with the default of green display with a REQ=NOADD option.

If the DDF status contains a CLEAR=Y, REQ=NOADD, and DDF posts a resource at that status, DDF removes the existing descriptor and does not add the new descriptor. Thus, DDF no longer displays the resource name.

If DDF uses resource status for color determination, code both Virtual Telecommunications Access Method (VTAM) status and automation status. Use wildcard characters to reduce the number of entries required. The NETSTAT command for DDF update uses VTAM status when you define DDFREFRESH as Y in the ENVIRON SETUP DDF control file. The current network status output from NETSTAT primes DDF.
Dynamic Updates

DDF uses automation status for dynamic updates during network automation activity. If you define the NETSTAT CHKAUTO parameter as Y, AON checks the automation flags for the resource before adding them to DDF. If this check determines that automation is off for the resource, DDF does not add the resource.

For example, when DDF sets PU001 to an INACTV state, the logging routines scan the control file for the INA* entry for DDF and generate a DDFADD request with a priority of 150. PU001 is displayed in red on the DDF status panel. After DDF sets PU001 to an ACTIVE state, the logging routines scan the control file for the active state DDF entry and generate a DDFDEL request. DDF then deletes PU001 from the DDF status panel.

Problem Resources

If a program detects a warning message for LINE02 on CNM01, DDF issues a DDFADD command to add a status descriptor for LINE02. The status display for LINE02 on system CNM01 now indicates a problem with LINE02. If you specify the DDF PROPAGATE UP parameter, the CNM01 field also reflects this.

Occasionally, a more serious problem might arise. The routine that detects this problem updates DDF with a status descriptor of a lower priority number. Because DDF links status descriptors in order of priority, the LINE02 status now reflects the status descriptor color of the more serious problem. After you resolve the more serious problem, the program detecting the resolution of the problem issues a DDFDEL command to remove LINE02 from the display.

Defining Status Panels

AON defines DDF status statements for status panels, in the CNMPNL1 data set and embeds the definitions in the EZLPNLS member with %INCLUDE. However, the DDF formats and internally builds the Detail Status Display.

If you define status components in the panel definitions, also define them in the corresponding tree structure. However, not all the status components defined in the tree structure require a corresponding entry on the Detail Status Display. In Figure 8 on page 18 the NETWORK status component is only a pseudo entry and is not defined on any DDF Detail Status Display.

You can customize the DDF status panels to reflect any environment. For example, you can define a panel to show the status of all the networks on all CPUs within the network operations center. The network operator then views the panel to determine the status of any of the network resources in the complex.

Defining Multiple Systems

You can define multiple NetView domains to DDF. You must first implement AON focal-point services for the target system DDF status update on the focal-point DDF. For more information on focal-points, refer to Tivoli NetView for z/OS Administration Reference.

In a multidomain environment, define a tree structure for each domain in the EZLTREE member of the DSIPARM data set on the designated focal-point DDF. Provide a unique root name for each systems tree structure, and match the focal-point root name with the ENVIRON SETUP SYSNAME entry in the control file. Also, code an entry for the SYSNAME parameter.
Because each root name is unique in a multiple systems environment, DDF uniquely addresses any status component on a system defined to the focal-point DDF. By adding the status component as a prefix to the root component name, DDF performs this addressing as the following example shows:

```
ROOT_COMPONENT.STATUS_COMPONENT (for example, CNM01.PU)
```

Similarly, AON routines add a prefix to DDF status descriptors shipped from the target system to the focal-point DDF with the root name of the target system.

---

**Implementing DDF**

When defining resources to DDF, you can explicitly define every network, gateway, domain, and resource in the DDF tree and panels. However, this technique has a disadvantage. As the number of resources tracked by AON increases, DDF maintenance grows more complex. To simplify this maintenance, AON provides a generic implementation of DDF that requires you to define only the group identifier, not the individual resources. For example, if you display resources using the generic definitions, the DDF tree requires only resource types. If you use this method, it is not necessary to define all network resources or all resource types known to this system (for example, to the PU level).

**Using Generic Implementation**

Generic implementation takes advantage of three functions within DDF:

- Storing detail records by priority
- Adding detail records using the default status component
- Displaying records using the status descriptor number

Use the default status component when a request to ADD, DELETE, or QUERY for the actual resource name fails. For example, if you add a component such as CNM01.PU001(RESOURCE) when PU001 is not in the tree, DDF attempts to add the component to the generic component of RESOURCE.

The relative position of the detail record in the detail record chain under a component determines the status descriptor number. Each component consists of detail records that were added directly to that component and all detail records with subordinate levels. For example, consider the following tree:

```
/* SYSTEM: CNM01 */
1 CNM01
  2 SYSTEM
    3 GATEWAY
    3 NETWORK
    4 RESOURCE
```

The DDFADD command for this tree is:

```
DDFADD CNM01.PU001(RESOURCE),IN=/PU001/
```

When you issue the DDFADD command, DDF adds the entry to the resource because PU001 is not defined in the tree.

When you request the detail records for the generic resource, DDF displays the detail records for all components added using the generic component RESOURCE and PU001. The DDFQRY command is:

```
DDFQRY CNM01.RESOURCE
```

Alternatively, you can code the panel like this:
When you request the detail records for NETWORK, DDF not only displays the detail records added directly to NETWORK as well as detail records added directly to generic component RESOURCE.

Alternatively, you can code the panel like this:

```
SF(CNM01.NETWORK,04,05,16,N, , ,nn)
```

DDFQRY CNM01.NETWORK

DDF arranges detail records in priority order under each component.

Complete the following steps to define your network to DDF using the generic component:

1. Define the generic components to the tree as the following example shows:

   1 CNM01
   2 SYSTEM
   3 NETWORK
   4 RESOURCE

2. Use the generic component in the tree to display the detail records at the specified positions on the panel. The empty status text field, coded as ST( ), causes DDF to display the INFO field of the DDFADD command for the detail record. If a descriptor with a descriptor number of 11 does not exist, DDF shows only 10 records and the status text field displays is empty.

   With this method, resources display in order of the priority of the condition of the resource. For example, if you use the DDF statements in the EZLCFG01 sample, DDF assigns the highest priority to red and the lowest to green. The following example shows a panel displaying a status of 12 network resources. This panel displays the highest priority error in the upper left position and the resources are listed in rows across the panel:

   ```
   /* DEFINE CNM01 NETWORK STATUS PANEL */
   P(EZLPNL1,24,80,SYSTEM,SYSTEM, , , )
   TF(01,27,57,WHITE,NORMAL)
   TT(CNM01 NETWORK STATUS)
   SF(CNM01.NETWORK,04,05,16,N, , ,01)
   ST( )
   SF(CNM01.NETWORK,04,25,36,N, , ,02)
   ST( )
   SF(CNM01.NETWORK,04,45,56,N, , ,03)
   ST( )
   SF(CNM01.NETWORK,04,65,76,N, , ,04)
   ST( )
   SF(CNM01.NETWORK,06,05,16,N, , ,05)
   ST( )
   SF(CNM01.NETWORK,06,25,36,N, , ,06)
   ST( )
   SF(CNM01.NETWORK,06,45,56,N, , ,07)
   ST( )
   SF(CNM01.NETWORK,06,65,76,N, , ,08)
   ST( )
   SF(CNM01.NETWORK,08,05,16,N, , ,09)
   ST( )
   SF(CNM01.NETWORK,08,25,36,N, , ,10)
   ST( )
   SF(CNM01.NETWORK,08,45,56,N, , ,11)
   ST( )
   SF(CNM01.NETWORK,08,65,76,N, , ,12)
   ST( )
   TF(24,01,48,T,NORMAL)
   ```

This technique gives you the advantage of first seeing the list of resources with the most critical problems. DDF determines resource panel position according to the order and priority of the problems. AON ships examples of components using this feature in the following members of CNMNL1:

- EZLPNLNS
- EZLPNLTY
- EZLPNLST
- EZLPNLI
- FKVPNL21

With DDF you can define generic values for your environment. Multiple generic values can apply to one message added to DDF to help organize network data in DDF. For example, you can test the RESTYPE message processing field to see whether it contains a valid generic value for DDF on this system. AON checks the ENVIRON DDF DDFGENERIC statements in the control file to verify generic values.

The DDFGENERIC RESTYPE VALUE=(restype,...) statement defines valid generic values in the control file. Samples provided with AON categorize DDF by resource type. For each valid generic value, define a panel which can display status descriptors added under this generic. In this case, SF (CNM01.PU,04,05,16,N,...) displays components for the PU generic. SF(CNM01.NETWORK,14,05,16,N,...) also displays components for the PU generic because the tree hierarchy shows that NETWORK is higher than PU.

**Specific Implementation of DDF**

If you define your network to DDF with the specific method, DDF displays resource names in a fixed position on the DDF panels. When DDF records a detail record for the resource, a resource name changes color. When no detail records exist for the resource, DDF shows the resource name in the status text field as blue, which is considered an empty color.

**Note:** When using the specific implementation of DDF, do not use REQ=NOADD in any DDF status definition in the control file.

Because large networks require excessive effort and storage to define and maintain in DDF, reserve specific implementation for small networks.

Complete the following steps to define your network to DDF using the specific component:

1. Define resources to the tree as shown in the following example:

```c
/* NETWORK: CNM01
1 CNM01
  2 SYSTEM
  3 GATEWAY
  3 NETWORK
  4 RESOURCE
  4 NCP01
  4 LINE01
```
2. Define resources to the panel as shown in the following example:

```c
/* DEFINE CNM01 NETWORK STATUS PANEL */
P(EZLPNL, 24, 80, SYSTEM, SYSTEM,,)
TF(01, 27, 47, WHITE, NORMAL)
TT(CNM01 NETWORK STATUS)
SF(CNM01.NCP01, 04, 05, 16, N,,)
ST(NCP01)
SF(CNM01.LINE01, 06, 05, 16, N,,)
ST(LINE01)
SF(CNM01.LINE02, 08, 05, 16, N,,)
ST(LINE02)
SF(CNM01.PU001, 10, 05, 16, N,,)
ST(PU001)
SF(CNM01.PU02, 12, 05, 16, N,,)
ST(PU02)
SF(CNM01.630-S, 14, 05, 16, N,,)
ST(630-S)
TF(24, 01, 48, T, NORMAL)
TT(PF1=HELP 2=DETAIL 3=END 6=ROLL 7=UP 8=DN)
TF(24, 51, 79, T, NORMAL)
TT(9=DEL 10=LF 11=RT 12=TOP)
EP
```

If you define the code as shown in the previous example, the DDF panel will look similar to [Figure 10](#). The resources are hard coded on the panel, so they are displayed at all times regardless of the status of the resources. This implementation differs from the default DDF implementation, which displays only resources with an exception status.

![Figure 10. Sample Panel Using Specific Implementation of DDF](#)

**Defining the Contents of DDF**

Table 5 on page 27 shows the elements that comprise DDF:
### Table 5. Defining DDF Contents

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>EZLTDDF</td>
<td>Task</td>
<td>Initializes DDF and maintains status information.</td>
</tr>
<tr>
<td>DDF</td>
<td>Command</td>
<td>Enters a DDF operator session.</td>
</tr>
<tr>
<td>DDFADD</td>
<td>Command</td>
<td>Adds status information.</td>
</tr>
<tr>
<td>DDFDEL</td>
<td>Command</td>
<td>Deletes status information.</td>
</tr>
<tr>
<td>DDFQRY</td>
<td>Command</td>
<td>Queries information.</td>
</tr>
<tr>
<td>DDFTREE</td>
<td>Command</td>
<td>Dynamically loads a tree member from the DSIPARM data set.</td>
</tr>
<tr>
<td>DDFPANEL</td>
<td>Command</td>
<td>Dynamically loads a panel member from the CNMPNL1 data set.</td>
</tr>
<tr>
<td>EZLINIT</td>
<td>Input file</td>
<td>Contains the initialization parameters defined with the statements described in &quot;Defining Initialization Statements (EZLINIT)&quot; on page 40. The EZLINIT member is in the DSIPARM data set.</td>
</tr>
<tr>
<td>EZLTREE</td>
<td>Input file</td>
<td>Contains the tree structures as described in &quot;Defining the Panel Hierarchy (EZLTREE)&quot; on page 38. This member can consist of a list of %INCLUDE statements that reference other members that contain tree structures. The format is %INCLUDE member, where member is the name of the member to include. The EZLTREE member is in the DSIPARM data set.</td>
</tr>
<tr>
<td>EZLPNLS</td>
<td>Input file</td>
<td>Contains the DDF panel parameters defined with the statements described in &quot;Defining the Panel Statements (EZLPNLS)&quot; on page 57. This member can consist of a list of %INCLUDE statements that reference other members that contain panel definitions. The format is %INCLUDE member, where member is the name of the member to include. EZLPNLS is in the CNMPNL1 data set.</td>
</tr>
<tr>
<td>panel_name</td>
<td>Input file</td>
<td>Contains the definition of a single panel. Name the member the panel name as defined with the PANEL statement.</td>
</tr>
<tr>
<td>tree_name</td>
<td>Input file</td>
<td>Contains the definition of a single tree structure. Name the member the root component name.</td>
</tr>
<tr>
<td>DDF</td>
<td>Control file entry</td>
<td>Defines status priority.</td>
</tr>
<tr>
<td>ENVIRON DDF</td>
<td>Control file entry</td>
<td>Defines DDF setup defaults for the system.</td>
</tr>
<tr>
<td>DDFGeneric</td>
<td>Control file entry</td>
<td>Defines valid values for implementing a generic set of values for DDF.</td>
</tr>
</tbody>
</table>

### Starting and Stopping DDF

During AON initialization, the EZLTDDF task loads the members that define panel format, navigation, and tree structures. The EZLINIT member defines parameters that are common to all DDF panels as well as basic initialization specifications such as the screen size, default function keys, and the initial screen displayed when a DDF session starts.

The EZLPNLS member defines parameters unique to specific panels, and the EZLTREE member defines the tree structures. These two members consist of either tree structure and panel definitions for all DDF panels or %INCLUDE statements that point to members containing definitions of single panels or tree structures.
Using %INCLUDE statements enables you to reload all definition members besides EZLINIT, EZLTREE, and EZLPNLS without the DDFADD command. For more information, see "Loading Panels" and "Loading Tree Structures". To change panels or tree structure specifications defined in the EZLTREE and EZLPNLS members, stop DDF and restart it to load the new definitions.

To start DDF, issue one of the following commands from a NetView console:

- `START EZL DDF`
- `START TASK=EZLDDDF`

To stop DDF, issue one of the following commands from a NetView console:

- `STOP EZL DDF`
- `STOP TASK=EZLDDDF`

When you stop DDF, it loses all existing status descriptors because the computer stores them only in memory. If you define DDFREFRESH=YES in the ENVIRON SETUP statement of the EZLCFG01 member, you can use the NETSTAT command to access VTAM displays and to prime DDF with current network exception conditions when it initializes. If you do not set DDFREFRESH to YES, DDF updates the file when it detects new failures, and then processes and logs the MONIT intervals.

---

**Loading Panels**

Without restarting DDF, you can dynamically load panels using either of the following two methods:

- Use the DDFPANEL command. For more information about the DDFPANEL command, see the following section, "Loading Tree Structures" and "Loading a Panel Member (DDFPANEL)" on page 106.
- Use the PANEL statement. The PANEL statement parameters search for a panel that is not defined in the EZLPNLS member. DDF locates a member in the CNMPNL1 data set with the same name as the requested panel name. For more information about the PANEL statement and its parameters, see "Defining New Panels (PANEL)" on page 59.

---

**Loading Tree Structures**

To dynamically load tree structures with the DDFTREE command, a member must exist in the DSIPARM data set that is named the same as the panel name or the root component in the tree structure. DDF makes the panels loaded with the DDFPANEL command resident only. All others are loaded when an operator issues commands that calls for them. DDF does not use %INCLUDE statements.

For example, you can change the tree structure for the root component CNM01 and the panel named EZLPNL1. You can also define these elements in members other than the EZLTREE or EZLPNLS members. You can then use the following commands to load the new definitions:

- `DDFTREE EZLTREES,ADD`
- `DDFPANEL EZLPNL1,ADD`

These commands enable you to load a small number of panels during initialization and add or delete panel subsets when required. This can significantly decrease the number of panels that DDF recognizes at any one time.
When DDF loads a new tree and replaces the existing tree, DDF copies status descriptors with similar leaf names, in both trees, to the new tree. For more information, see “Loading Tree Members (DDFTREE)” on page 109.

Using the Definition Procedure

When you start a DDF session, use the following procedure to define the panels that DDF displays:

1. Define the component hierarchy as described in “Defining the Panel Hierarchy (EZLTREE)” on page 38. The EZLTREE tree contains tree structure definitions. Perform either or both of the following procedures:
   - Place all tree structure definitions in this member.
   - Use %INCLUDE statements to point to other members that contain definitions for entire tree structures.

If you use %INCLUDE statements when you add and delete trees, use the same name for referenced members and the root component. Complete the trees, starting at level 1. DDF deletes trees by the root name and adds them by the DSIPARM member name.

To load some tree structures during initialization and after DDF is started use the EZLTREE member to define those tree structure entries that you load during initialization. Then, use the DDFTREE command to load additional tree structures as needed. After you start DDF, put the tree structures in separate members. Name each member after the root component for which you define the tree structure.

The following example shows a typical tree structure definition for AON/SNA:

/* NETWORK : CNM01 */
1 CNM01
  2 SYSTEM
    3 GATEWAY
    3 GROUPS
      4 CALIF
      5 LA
      5 SANFRAN
      5 SANDIEGO
    4 NEWYORK
    4 ATLANTA
  3 OPID
    4 OPER1
  3 NETWORK
    4 RESOURCE
    4 SNA
      5 SA
      6 NCP
      6 LINE
      6 LINKSTA
      6 CDRM
      6 CORSC
      6 PU
      6 LU
      6 SESSION
      6 APPL
      6 ERR
    5 APPN
      6 CP
      6 EN
    5 X25
      6 X25MCH
      6 X25PU
In the preceding example, CNM01 is the root component. Embed this definition in the EZLTREE member or in a separate member named CNM01. "Defining the Panel Hierarchy (EZLTREE)" on page 38 describes tree structures in detail.

Define the initialization and common panel specifications as described in "Defining Initialization Statements (EZLINIT)" on page 40. The EZLINIT member defines parameters that are common to all DDF panels and basic initialization specifications such as:

- Screen size
- Initial screen shown when you start the DDF session
- Maximum operator logon limit
- Temporary error limit value
- Default function key definitions
- Detail function key definitions
- Detail function key descriptions
- Default priorities and colors

Define the panels as described in "Defining the Panel Statements (EZLPNLS)" on page 57. After you initialize DDF, use the EZLPNLS member to load the panel definitions. This member can contain the following:

- Definitions for all panels.
- %INCLUDE statements that point to separate members containing panel definitions.
- Combination of both panel definitions and %INCLUDE statements.
- Subset of panel entries that are loaded during initialization so that you can load additional panel definitions when needed. For more information, see "Loading Panels" on page 28.

The default 3270 screen size for DDF is 24 rows by 80 columns. The VTAM LOGMODE used for any DDF (NetView) 3270 work station must specify the default screen size of 24 x 80. The alternate screen size in the LOGMODE can specify any size supported by NetView. DDF uses the 3270 command Erase/Write to present its screens to the user, which uses the default screen size specified in the LOGMODE.

The following example shows a typical menu panel definition named EZLPNLST. Embed this definition in the EZLPNLS member or in a separate member named EZLPLNST.
Table 6 provides a description for each statement in the preceding example.

**Table 6. Menu Panel Definitions**

<table>
<thead>
<tr>
<th>Statement and Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>P(EZLPNLST,24,80,,,,,)</td>
</tr>
</tbody>
</table>

This is the panel definition statement. The panel name is EZLPNLST, the length of the panel is 24, and the width of the panel is 80.

<table>
<thead>
<tr>
<th>TF(01,02,09,T,NORMAL)</th>
</tr>
</thead>
</table>

This is the text location statement used to define constant panel fields. This field starts on line 01, in position 02, and ends in position 09. The color of the field is white and the highlighting is normal.

<table>
<thead>
<tr>
<th>TT(EZLPNLST)</th>
</tr>
</thead>
</table>

The text data statement (EXLPNLST) that specifies data that goes in the field just defined.

<table>
<thead>
<tr>
<th>TF(02,25,57,WHITE,NORMAL)</th>
</tr>
</thead>
</table>

Text location statement for another constant field.

<table>
<thead>
<tr>
<th>TT(DATA CENTER NETWORKS)</th>
</tr>
</thead>
</table>

Text data statement for the field just defined. Notice that TF and TT are grouped in pairs.

<table>
<thead>
<tr>
<th>SF(CNM01.NETWORK,04,05,10,N,,EZLPNL2)</th>
</tr>
</thead>
</table>

This statement defines the location of the status component field. The status component is CNM01, this field starts on line 04 in position 05 and ends in position 10. The highlight level is normal, and the next panel displayed when F8 (Down) is pressed is EZLPNL2.

<table>
<thead>
<tr>
<th>ST(CNM01)</th>
</tr>
</thead>
</table>

This text is displayed in the SF field. The field name is CNM01. Notice also that SF and ST are grouped in pairs.
Table 6. Menu Panel Definitions (continued)

<table>
<thead>
<tr>
<th>Statement and Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SF(CNM01.SYSTEM,04,17,30,N,,EZLPNL01)</td>
</tr>
<tr>
<td>Status field definition.</td>
</tr>
<tr>
<td>ST(MESSAGEVIEW)</td>
</tr>
<tr>
<td>Status text definition.</td>
</tr>
<tr>
<td>SF(CNM01.GATEWAY,04,65,72,N,,EZLPNLG)</td>
</tr>
<tr>
<td>Status field definition.</td>
</tr>
<tr>
<td>ST(GATEWAY)</td>
</tr>
<tr>
<td>Status text definition.</td>
</tr>
<tr>
<td>TF(24,01,39,T,NORMAL)</td>
</tr>
<tr>
<td>TT(1=HELP 2=DETAIL 3=RET 6=ROLL 8=DN)</td>
</tr>
<tr>
<td>TF(24,40,79,T,NORMAL)</td>
</tr>
<tr>
<td>TT(10=LF 11=RT 12=TOP)</td>
</tr>
</tbody>
</table>

Here, TF and TT are used to display the function key definitions, which these are the defaults defined in EZLINIT. You can define function keys unique to this panel.

EP

This is the end panel statement used to indicate that this is the end of definitions for this panel.

---

Figure 12 shows the panel that is displayed when you run the statements listed in Table 6 on page 31.

Figure 12. Data Center Networks (EZLPNLST) Panel

EZLPNLST

DATA CENTER NETWORKS

<table>
<thead>
<tr>
<th>CNM01</th>
<th>MESSAGEVIEW</th>
<th>GROUPS</th>
<th>OPERATORS</th>
<th>GATEWAY</th>
</tr>
</thead>
</table>

06/24/00 16:21:00

PF1=HELP 2=DETAIL 3=END 6=ROLL 7=UP 8=DN 10=LF 11=RT 12=TOP
Modifying the Control File for DDF

The EZLCFG01 sample control file requires minimal changes to set up DDF. You must, however, customize the definition of your NetView domain. Customize ENVIRON SETUP SYSNAME=CNM01 to reflect the system name you choose for AON. You should use the NetView domain ID. The value is referred to as the SYSNAME in this document. The SYSNAME value is used during panel and tree installation.

In the sample implementation, the ENVIRON DDF statement is defined as follows:

```
ENVIRON DDF DDF=STATUS,DDFREFRESH=NO,DDFGENERIC=(RESTYPE),DDFAUTO=NO
```

Where:

**DDF=STATUS**
Causes the resource status to determine the color of the indicators in DDF.

**DDFREFRESH=NO**
Causes DDF to not be taken from current VTAM status only at DDF initialization. In some networks, the displays required to do this can cause an increase in CPU usage during initialization. If you want to do this, change this definition to DDFREFRESH=YES.

**DDFGENERIC=(RESTYPE)**
Causes DDF to group the resource status descriptors and display them by resource type.

**DDFAUTO=NO**
Causes DDF to display status changes for the resources regardless of whether the automation flag is on. If you do not want failures of non-automated resources posted to DDF, change this definition to DDFAUTO=YES.

Other control file definitions that influence DDF are the DDF and DDFGENERIC statements. You should use the AON defaults for these definitions.

Modifying the EZLTREE Tree Structure

Tree structures define resources into hierarchical groups, with the domain as the highest component in the hierarchy.

The sample tree provided with AON enables DDF to display current problems in your networking environment, with minimal maintenance. Change domain CNM01 in the EZLTREE member to reflect your SYSNAME value (domain ID).

Modifying DDF Panels

The sample panels copied into the CNMPNL1 data set during installation set up a typical implementation of DDF. To activate DDF, edit the following panels, changing all occurrences of CNM01 to your domain ID (SYSNAME value on ENVIRON SETUP statement):

- EZLPNLG
- EZLPNLOA
- EZLPNLO2
- EZLPNL1
- EZLPNLG1
- EZLPNLOB
- EZPNLST
- EZLPNL2
- EZLPNL1
- EZLPNL1
- EZLPNL1
- EZLPNL1
- EZLPNL1
- EZLPNL1
If you want to use the sample DDF group, you need to change the domain ID in the following panels:

<table>
<thead>
<tr>
<th>Panel 1</th>
<th>Panel 2</th>
<th>Panel 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>EZLPNLGR</td>
<td>EZLPNLIA</td>
<td>EZLPNLNY</td>
</tr>
<tr>
<td>EZLPNLSD</td>
<td>EZLPNLSD</td>
<td>EZLPNLAT</td>
</tr>
<tr>
<td>EZLPNLCA</td>
<td>EZLPNLCA1</td>
<td></td>
</tr>
</tbody>
</table>

To use the same operator signout panels, change the domain ID in the following panels:

<table>
<thead>
<tr>
<th>Panel 1</th>
<th>Panel 2</th>
<th>Panel 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>EZLPNLW0</td>
<td>EZLPNLW1</td>
<td>EZLPNLW2</td>
</tr>
</tbody>
</table>

---

**Adding AON/TCP and SNA to the Main DDF Panel**

Complete the following sections to add the AON components to the main DDF panel.

### Adding TCP to the Main DDF Panel

To have a TCP/IP selection appear on the main DDF panel, do one of the following:

- If you are using the EZLPNLST member as your main DDF panel (the first panel to display), add the following statements:
  
  ```
  SF (CNM01.TCPIP,10,10,17,N, ,FKXPNLT)
  ST (TCPIP)
  ```

  These statements are also in the FKXPNLST data set and you can copy them from there.

- If you are using the EZLPNLTY panel as your main DDF panel, add the following statements:
  
  ```
  SF (CNM01.TCPIP,10,10,17,N, ,FKXPNLT7)
  ST (TCPIP)
  ```

  These statements are also in the FKXPNLTY data set and you can copy them from there.

**Note:** Remember to update CNM01 to your domain ID.

### Adding SNA to the Main DDF Panel

To use AON/SNA, add AON/SNA to the main DDF panel by doing one of the following:

- If you are using the EZLPNLST panel as your main DDF panel (the first panel to display), add the following statements:
  
  ```
  SF (CNM01.NETWORK,06,10,15,N, ,FKVPNSNA)
  ST (SNA01)
  ```

  These statements are also in the FKVPNLST data set and you can copy them from there.

- If you are using EZLPNLTY as your main DDF panel, add the following statements:
  
  ```
  SF (CNM01.NETWORK,06,10,15,N, ,FKVPNL1)
  ST (SNA)
  ```
These statements are also in the FKVPNLTY data set and you can copy them from there.

**Editing EZLPNLS**

To customize the panel list, copy the following members into the bottom of EZLPNLS:

- FXXPNLS (TCP)
- FKVPNLS (SNA)
Chapter 3. Defining Dynamic Display Facility (DDF) Statements

Table 7 lists the members where Dynamic Display Facility (DDF) statements are defined. This figure also explains the purpose of each member and shows where to find more information about each member.

Table 7: Members Containing DDF Statements

<table>
<thead>
<tr>
<th>Member</th>
<th>Purpose</th>
<th>Location of Heading</th>
</tr>
</thead>
<tbody>
<tr>
<td>EZLTREE</td>
<td>Defines panel hierarchy</td>
<td>“Defining the Panel Hierarchy (EZLTREE)” on page 38</td>
</tr>
<tr>
<td>EZLINIT</td>
<td>Defines initialization statements</td>
<td>“Defining Initialization Statements (EZLINIT)” on page 40</td>
</tr>
<tr>
<td>EZLPNLS</td>
<td>Defines panel statements</td>
<td>“Defining the Panel Statements (EZLPNLS)” on page 53</td>
</tr>
<tr>
<td>EZLCFG01</td>
<td>Defines generics and status color relationships.</td>
<td>Refer to the Tivoli NetView for z/OS Administration Reference for more information.</td>
</tr>
</tbody>
</table>
Defining the Panel Hierarchy (EZLTREE)

Purpose
The hierarchy for status color changes are defined in the EZLTREE member in the DSIPARM data set. When the status changes for a component, the corresponding color change extends up or down the tree to the higher or lower level components. The level number assigned to each component determines the level. The entry in the EZLINIT member or the individual DDFADD command requests determines the type of propagation.

The following syntax diagram shows the level number entry.

Syntax

```
level_number
```

Parameters

- **level_number**
  Defines a valid number between 1 and 99. A tree starts with a root level of 1.

- **status_component**
  Defines the resource for which status information is displayed. AON uses the resource name as defined in the control file. The status component entry for the root matches the ENVIRON SETUP SYSNAME entry in the control file.

- **empty_chain_color**
  Defines a color for the empty chain. You can use red, blue, turquoise, pink, green, yellow, or white. If DDF does not associate a status descriptor with a status component, the status component is displayed in this color on the DDF status panel. This entry is optional and can also be coded in the EMPTYCOLOR entry of the EZLINIT member. See “Defining Default Colors for Status Components (EMPTYCOLOR)” on page 55 for more details.

- %INCLUDE member
  Includes members that contain EZLTREE statements, where member is the member name. Each included member must start with level 1 and must contain an entire tree structure.

Usage

The level numbers define the order of dependence. For an example, see Figure 13 on page 35. In that figure, NCP is defined to be dependent on NETWORK. Therefore, when a change occurs for an NCP, it is reflected on NCP, NETWORK, SYSTEM, and CNM01.

You cannot use duplicate status components within the same tree. Not all status components that are defined in the tree need to have a corresponding panel entry or a status descriptor associated with them. Each root name must be unique to avoid addressing conflicts. DDF can address each status component defined in the tree as follows:
Example

Two separate AON/SNA trees, CNM01 and CNM02, representing two different networks are defined [Figure 13]. In this example, CNM01 is the focal point and CNM02 is the remote domain. In the EZLTREE member on CNM01, define a tree for both CNM01 and CNM02.

Both trees start with level number 1 and each has a unique root name. They can have similar status component names, such as LINE, PU, and NCP. The corresponding entry for the root component in the control file on system CNM01 is:

```
environ setup,
   netview=net,
   sysname=cnm01,
```

Figure 13. DDF Tree Structure

Both trees start with level number 1 and each has a unique root name. They can have similar status component names, such as LINE, PU, and NCP. The corresponding entry for the root component in the control file on system CNM01 is:

```
environ setup,
   netview=net,
   sysname=cnm01,
```
Do not define the root component name CNM02 on the ENVIRON SETUP SYSNAME keyword in the control file for CNM01. Instead, define it in a similar way in the ENVIRON SETUP SYSNAME keyword in the control file for CNM02. Assuming that AON notification forwarding is established between CNM02 and CNM01, DDF adds a CNM02 prefix to all status components when status information is forwarded to CNM01. For example, it adds CNM02.PU01 for PU01. The EZLTREE member in the CNM02 system contains only the tree for CNM02.

### Defining Initialization Statements (EZLINIT)

The EZLINIT member in the DSIPARM data set defines the DDF initialization parameters. You should use the defaults supplied in the EZLINIT member in DSIPARM.
Defining the Screen size (SCREENSZ)

**Purpose**
The SCREEN SIZE parameter defines the screen buffer size. This entry is optional. If you do not specify this parameter, AON uses a program default value of 3000.

The following syntax diagram show the screen size parameter.

**Syntax**

```
SCREENSZ
```  

**Parameters**

`number`
Values can be in the range of 3000–9999.

**Usage**
For a larger screen buffer size, increase the `number` parameter.

**Example**

```
SCREENSZ = 4000
```
Linking the Chain Detail Records (CHAIN)

**Purpose**
The CHAIN parameter defines whether the detail status is linked in descending order (last to most recent) or in ascending order (most recent to last).

The following syntax diagram shows the chain parameter.

**Syntax**
CHAIN

**Parameters**
- **A**: Link detail records within the same priority in ascending order.
- **D**: Link detail records within the same priority in descending order.

**Example**
CHAIN = A
Defining the Initial Screen (INITSCRN)

**Purpose**
The INITSCRN parameter defines the initial panel that is displayed by DDF.

**Syntax**

```
INITSCRN

INITSCRN=panel
```

**Parameters**

- **panel**
  
  A valid alphanumeric name with maximum length of eight characters.

**Usage**

If you change the initial panel as defined in the EZLPNLST member of the DSIPARM data set, also change the name in the INITSCRN entry.

**Example**

```
INITSCRN = EZLPNLST
```
Defining the Number of Operators (MAXOPS)

Purpose
The MAXOPS parameter defines the maximum number of logged on operators that can use DDF. This entry is optional. If you do not specify this parameter, the default is 30.

Syntax
MAXOPS = number

Parameters
number
Values range from 1 to 99.

Usage
If the number of operators attempting to use DDF is more than the number defined in the MAXOPS parameter, AON denies the additional operators access to DDF. The dynamic update facility keeps an internal count of the operators that are logged on.

Example
MAXOPS = 35
Propagating Status Upward (PROPUP)

Purpose
The PROPAGATE UP parameter defines whether status word text should be sent up the status tree as a system default. This entry is optional. If you do not specify this parameter, the default is YES.

Syntax
PROPUP

Usage
Specify a value of YES. This parameter can be overridden with individual DDFADD requests. See "Adding Status Descriptors (DDFADD)" on page 99 for more information.
Propagating Status Downward (PROPDOWN)

Purpose
The PROPAGATE DOWN parameter defines whether status information should be sent down the status tree as a system default. If you do not specify a value, the default is NO. This entry is optional.

Syntax
PROPDOWN

Usage
Specify a value of NO. This parameter can be overridden with individual DDFADD requests. See "Adding Status Descriptors (DDFADD)" on page 99 for more information. This entry is optional. If not coded, the default is NO.
Defining the Temporary Error Limit (TEMPERR)

Purpose
The TEMPERR parameter defines the maximum number of temporary input and output errors you can receive when trying to display a DDF panel. This entry is optional. If you do not specify a value, the default is 3.

Syntax
TEMPERR

Parameters
number
Values range from 3 to 99.

Usage
Use the value 5, which is provided in EZLINIT member in the DSIPARM data set.

Example
TEMPERR = 5
Defining the Default Function Key Definitions (PFKnn)

Purpose
The PFKnn parameter defines the default function key settings. You cannot change F3.

Syntax
PFKnn command var

Parameters
nn  Values range from 1 to 24.
command  The command is issued when the function key is pressed.
var  A variable or any text data passed with this command. Use the following variables as part of the command:
  &COMP  Identifies the component.
  &COMPAPPL  Identifies the generic component within parentheses (for example, PU01(PU)). The generic component is shown only if the component was added using one.
  &ROOT  Identifies the root or system.
  &SYSDATE  Identifies the system date.
  &SYSTIME  Identifies the system time.
  &IN or &INFO  Identifies the detail entry word text that is displayed on the status screen.
  &DATE  Identifies the date that the detail entry was added.
  &TIME  Identifies the time that the detail entry was added.
  &SENDERID  Identifies the reporter who submitted the detail entry.
  &SNODE or &SENDERNODE  Identifies the node of the reporter who submitted the detail entry.
  &DA or &DATA  Identifies the actual message text.
  &RV or &REFVALUE  Identifies the reference value of the detail entry.
  &PR or &PRIORITY  Identifies the priority of the detail entry.
&CO or &COLOR
Identifies to the color of the detail entry.

&HI or &HIGHLITE
Identifies the highlight level of the detail entry.

Usage
You can redefine all of the function keys except F3.

Example
In this example, the DIS PU01 command is issued when F4 is pressed while the cursor is placed on the PU01 entry on the status screen:

PF4=DIS &INFO
Defining the Detail Function Key for the Detail Display
(DPFKnn)

Purpose
The DPFKnn parameter defines the function keys that are unique to the detail panel.

Note: The function keys defined are active only when the detail panel is displayed and, therefore, override the default settings defined with the PFKnn statement.

Syntax

\[ \text{DPFK} \]

\[ \text{DPFKnn=command var} \]

Parameters

\( nn \)  Values range from 1 to 24.

\( command \)

The command issued when the defined function key is pressed.

\( var \)

The variables passed along with the command when the defined function key is pressed.

The following variables can be used as the command:

\&COMP
Identifies the component.

\&COMPAPPL
Identifies the generic component within parentheses (for example, PU01(PU)). The generic component is shown only if the component was added by using the value of 1.

\&ROOT
Identifies the root or system.

\&SYSDATE
Identifies the system date.

\&SYSTIME
Identifies the system time.

\&IN or \&INFO
Identifies the detail entry word text that is displayed on the status panel.

\&DATE
Identifies the date the detail entry was added.

\&TIME
Identifies the time the detail entry was added.

\&SENDERID
Identifies the reporter who submitted the detail entry.

\&SNODE or \&SENDERNODE
Identifies the node of the reporter who submitted the detail entry.
&DA or &DATA
Identifies actual message text.

&RV or &REFVALUE
Identifies the reference value of the detail entry.

&PR or &PRIORITY
Identifies the priority of the detail entry.

&CO or &COLOR
Identifies the color of the detail entry.

&HI or &HIGHLITE
Identifies the highlight level of the detail entry.

Usage
You can tailor all of the functions except F3.
Describing Function Keys on the Detail Panel, Part 1
(DPFKDESC1)

Purpose
The DPFKDESC1 parameter defines the first part of the function key description
that is displayed at the bottom of the detail panel. This text is concatenated with
the text defined with the DPFKDESC2 statement.

Syntax
DPFKDESC1

Parameters

text
Up to 40 characters of data.

Example
DPFKDESC1='PF1=HELP PF2=END PF3=RETURN'
Describing Function-Key Text on the Detail Panel, Part 2 (DPFKDESC2)

**Purpose**
The DPFKDESC2 parameter defines the second part of the PF key description that is displayed at the bottom of the detail display. This text is concatenated with the text defined with the DPFKDESC1 statement.

**Syntax**

```plaintext
DPFKDESC2
```

```plaintext
DPFKDESC2=text
```

**Parameters**

- `text`
  Up to 40 characters of data.

**Example**

```plaintext
DPFKDESC2='PF6=ROLL PF7=UP PF8=DOWN'
```
Defining Default Colors (DCOLOR)

Purpose
The DCOLOR parameter defines the appropriate color for a status descriptor that is outside of the defined priority and color ranges. This entry is optional. If you do not specify a value, the default is green.

Syntax
DCOLOR

Parameters
color
Blue, red, pink, green, turquoise, yellow, and white are valid colors.

Usage
Use the value WHITE, which is provided in the EZLINIT member in DSIPARM, it does not conflict with existing status and color definitions.

Example
DCOLOR = WHITE
Defining Default Colors for Status Components
(EMPTYCOLOR)

**Purpose**
The EMPTY COLOR parameter defines the color that should be displayed for a status component that has no status descriptor associated with it. This entry is optional. If you do not specify a value, the default is green.

**Syntax**
EMPTYCOLOR

**Parameters**
color
Blue, red, pink, green, turquoise, yellow, and white are valid colors.

**Usage**
Use the color BLUE, which is provided in the SEZLSENU member of EZLINIT, it does not conflict with existing status or color definitions. This parameter can be overridden in the EZLTREE member.

**Example**
EMPTYCOLOR = BLUE
Defining Priority and Color Ranges (PRITBLSZ)

**Purpose**
The PRITBLSZ parameter defines the number of priority and color ranges defined by the PRIORITY entries. This entry is optional. If you do not specify a value, the default is 7.

**Syntax**
PRITBLSZ

**Parameters**

nn Valid number greater than 7

**Usage**
Use the number 12, which is provided with the EZLINIT member in DSIPARM.

**Example**
PRITBLSZ = 12
Defining Color and Priority Ranges (PRIORITY)

Purpose
The PRIORITY parameter defines the relationship between colors and priority ranges. This entry is optional. If you do not specify values, the defaults are:

<table>
<thead>
<tr>
<th>Color</th>
<th>Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>Red</td>
<td>001 to 099</td>
</tr>
<tr>
<td>Pink</td>
<td>100 to 199</td>
</tr>
<tr>
<td>Yellow</td>
<td>200 to 299</td>
</tr>
<tr>
<td>Blue</td>
<td>300 to 399</td>
</tr>
<tr>
<td>Turquoise</td>
<td>400 to 499</td>
</tr>
<tr>
<td>White</td>
<td>500 to 599</td>
</tr>
<tr>
<td>Green</td>
<td>600 to 699</td>
</tr>
</tbody>
</table>

Syntax
PRIORITY

Parameters

nnn
Valid number between 001 and 999.

mmm
Valid number between 001 and 999 and equal to or greater than the value specified in nnn.

color
Red, green, yellow, turquoise, pink, blue, and white are valid colors.

Usage
Use the values provided with the EZLINIT member in DSIPARM.

Example

PRIORITY=100,199,RED
PRIORITY=200,299,PINK
PRIORITY=300,399,YELLOW
PRIORITY=400,499,TURQUOISE
PRIORITY=500,599,GREEN
PRIORITY=600,699,BLUE
DCOLOR=WHITE
EMPTYCOLOR=BLUE

Defining the Panel Statements (EZLPNLS)

The EZLPNLS member in the CNMPNL1 data set defines the DDF status panels. The structure of each panel definition is as follows:

1. Beginning panel definition (PANEL)
2. Status component definition (STATUSFIELD and STATUSTEXT)
3. Text fields and data definition (TEXTTEXT and TEXTFIELD)
4. Function key definition unique to this panel (PFKn)
5. End panel definition (ENDPANEL)
Defining New Panels (PANEL)

Purpose
The panel entry identifies the start of a new panel and its general attributes.

Syntax

```
PANEL (name, length, width, top_panel, up_panel)
```

Parameters

- **name**
  Identifies a user-defined panel name up to eight characters in length.

- **length**
  Defines the number of lines or rows on the screen. You must specify a numeric value.

- **width**
  Defines the number of columns on the screen. You must specify a numeric value.

- **top_panel**
  Defines the panel that displays when the TOP command is issued or the appropriate function key is pressed.

- **up_panel**
  Defines the panel that displays when the UP command is issued or the appropriate function key is pressed.

- **down_panel**
  Defines the panel that displays when the DOWN command is issued or the appropriate function key is pressed.

- **left_panel**
  Defines the panel that displays when left panel function key is pressed or LEFT command is issued.

- **right_panel**
  Defines the panel that displays when the right panel function key is pressed or the RIGHT command is issued.

Usage
The default initial panel name supplied with AON is called EZLPNLST. If you change this name, also change the definition in the EZLINIT member for the INITSCRN entry to reflect this. If you have more data than can be displayed on a single screen, you can define continuation panels with the `left_panel_name`, `right_panel_name`, and `down_panel_name` parameters.

The parameters are positional.
Example
This example defines EZLPNSNA as the panel name:

```
PANEL(EZLPNSNA,24,80,EZLPNL1,EZLPNL1, , ,)
```

The length of the panel is 24 and the width is 80. The panel named EZLPNL1 is displayed when the TOP command is used and the panel named EZLPNL1 is displayed when the UP command is used. No entries are defined for the DOWN, LEFT, or RIGHT commands.
Locating the Status Component (STATUSFIELD)

Purpose
The STATUSFIELD entry defines the location of the status component on a panel and the panels that display when the UP and DOWN commands are used. A STATUSFIELD entry is always accompanied by a STATUSTEXT entry.

Syntax

```plaintext
STATUSFIELD
```

```
(root_cmpt, status_cmpt), start_line
```

```
, start_position, end_position, highlight, up_panel
```

```
, down_panel, status_descriptor_number
```

Parameters

**root_cmpt**
Defines the root name as defined in the root node of the tree structure. You must specify the root (as opposed to the status component alone), because different systems can have status components with the same name defined in their respective tree structures. For example, the status component LINE is often used. Because the root and status component must always be unique, each status component in a tree structure can be uniquely identified by adding the root component entry as a prefix.

**status_cmpt**
Defines the status component name as defined in the EZLTREE member. The maximum length is eight characters.

**start_line**
Defines the line number where the status component should be displayed. Entry should be numeric and within the range specified in the length parameter in the panel definition statement.

**start_position**
Defines the actual column number on the start line. The start line is specified above where the component is to be placed. There must be a minimum of 2 spaces between the ending position of one field and the beginning position of the next field to allow for attribute type. For example, if the end position of the status field is in column 10, the start position of the next STATUSFIELD must be in column 13.

**Note:** You cannot start the component in position one because it always needs one leading byte.

**end_position**
Defines the column number in which the definition of the component ends. This is governed by the length of the text that is defined in the STATUSTEXT definition. For example, if you define LINE, the length of the STATUSTEXT is four and the end position is the start position plus three. See “Defining the
Text Area (STATUSTEXT) on page 63 for more information. You must have at least 2 bytes between the end position of one field and the beginning position of the next field for attribute bytes.

**highlight**
- Defines the type of highlighting. You can use normal, blink, reverse, or underscore. Define highlighting as normal is recommended, it can be overridden with individual DDFADD requests.

**up_panel**
- Defines the panel that is to be displayed if the Up function key is pressed.

**down_panel**
- Defines the panel that is to be displayed if the Down function key is pressed.

**status_descriptor_number**
- Defines the number of the status descriptor. Use a numeric value in the range of 1–999. Status descriptors are chained with the status component in ascending order of priority. A panel can be designed with the same status-component name defined in every STATUSFIELD entry. The status descriptor number specifies the status descriptor displayed in each field. For example, if you specified two STATUSFIELD entries for the same status component with status descriptor numbers 1 and 2, the status descriptor with the higher priority is displayed in the status field with status descriptor number 1. The next higher priority status descriptor is displayed in the status field with status descriptor number 2.

You can add a letter that denotes the type of information to be displayed as a prefix for the status descriptor. If you do not add a prefix, you can provide text with individual DDFADD requests using the INFO keyword. See Adding Status Descriptors (DDFADD) on page 94 for more information. Valid description prefixes are:
- **C** Displays the name of the component
- **D** Displays the date the record was added
- **M** Displays the message text
- **P** Displays the priority of the record
- **R** Displays the name of the root
- **S** Displays the ID of the requester
- **T** Displays the time the record was added
- **U** Displays the number of duplicate records
- **V** Displays the reference value of the requester
- **X** Displays the domain of the requester

**Usage**
- When designing a panel for any status component, make the end position greater than or equal to the start position; otherwise, unpredictable results may occur during DDF initialization.

The component start position should be column 02.

Parameters are positional.

**Example**
- In this example, the LINES on CNM01 status component start on line 04 in column 10, end in column 14, have normal highlighting:

```
STATUSFIELD(CNM01.LINES,04,10,14,NORMAL,,)
```

No entries are defined for the UP or DOWN commands.
Example
In this example, the status component starts on line 02 in column 04, ends in column 11, and has normal highlighting:
\[SF(CNM01.SYSTEM,02,04,11,N,,EZLPNSNA)\]

No entries are defined for the UP panel and the EZLPNSNA entry is defined for the DOWN command.

Example
In this example, three STATUSFIELD entries are defined for the same CNM01.PU status component:
\[SF(CNM01.PU,02,04,11,NORMAL, , ,01)\]
\[SF(CNM01.PU,03,04,11,NORMAL, , ,02)\]
\[SF(CNM01.PU,04,04,11,NORMAL, , ,03)\]

The highest priority status descriptor is displayed in the first entry, the next in the second, and so on.

Example
In this example, three STATUSFIELD entries are defined for the same CNM01.RESOURCE status component:
\[SF(CNM01.RESOURCE,04,02,05,NORMAL, , ,P01)\]
\[SF(CNM01.RESOURCE,04,08,15,NORMAL, , ,C01)\]
\[SF(CNM01.RESOURCE,04,18,79,NORMAL, , ,M01)\]

The first status descriptor is displayed, with its priority in column 02, its component name in column 08, and its message text starting in column 18.
Defining the Text Area (STATUSTEXT)

Purpose
The STATUSTEXT entry defines the text data displayed in the STATUSFIELD entry. The value of this entry is normally the name of the status.

Syntax

```
STATUSTEXT
```

Parameters

```
text
```
Defines the data displayed for the status component defined in STATUSFIELD entry. The maximum length is 76 alphanumeric characters. It is recommended that you use the name specified for the status component, such as CNM01.LINE. The length of text determines the end position that is coded in the STATUSFIELD entry.

Usage

Each STATUSFIELD entry must have a STATUSTEXT entry associated with it.

The STATUSTEXT text can be overridden by individual DDFADD requests using the INFO keyword. See "Adding Status Descriptors (DDFADD)" on page 99 for more information.

Example

In this example, the resource type LINE, on CNM01 displays as LINE on the status display panel:

```
STATUSFIELD(CNM01.LINES,04,10,13,NORMAL, ,)
STATUSTEXT(LINE)
```

Note: The end position in the STATUSFIELD reflects the length of resource line names.

Example

In this example, the text is not specified for the STATUSTEXT entry for LINE:

```
STATUSFIELD(CNM01.LINE,10,15,25,NORMAL, ,)
STATUSTEXT( )
```

You can use twelve characters for any text by the associated STATUSFIELD entry. Provide this text with individual DDFADD requests by using the INFO keyword.
Defining Text Location (TEXTFIELD)

Purpose
This entry defines the location and attributes of fields that remain constant such as panel headings, field names, and function key designations.

Syntax
TEXTFIELD
TEXTFIELD(—start_line—,start_position—,end_position—,color—,highlight—)

Parameters

start_line
Defines the line number on which the text field is displayed. Entry should be numeric and within the range specified in the length parameter in the panel definition statement.

start_position
Defines the column number on which the text field is placed.

dend_position
Defines the column number in which the data specified in entry TEXTTEXT ends. See “Defining Displayed Text (TEXTTEXT)” on page 65 for more details.

color
Defines the color of the text. You can use blue, red, green, yellow, pink, turquoise, or white for the color to be displayed in the TEXTTEXT entry text.

highlight
Defines the type of highlighting. You can use normal, blink, reverse, or underscore to determine the TEXTTEXT entry text display.

Usage
When designing a panel, for any TEXTFIELD, make the end position of the TEXTFIELD greater than, or equal to, the start position. Otherwise, unpredictable results may occur during DDF initialization.

The parameters are positional.

Example
In this example, the TEXTFIELD is defined as being on line 1, starting in column 25, ending in column 57, white in color, and with normal highlighting:
TEXTFIELD(01,25,57,WHITE,NORMAL)
Defining Displayed Text (TEXTTEXT)

**Purpose**
The TEXTTEXT entry defines the data displayed in the corresponding TEXTFIELD entry.

**Syntax**
TEXTTEXT

```
TEXTTEXT(text)
```

**Parameters**

*text*

Defines the data displayed for the TEXTFIELD entry. The length of the data determines the end position that is coded in the TEXTFIELD entry. Maximum length is 72 alphanumeric characters.

**Usage**
Each TEXTFIELD entry must have a TEXTTEXT entry associated with it.

**Example**
In this example, the text DATA CENTER NETWORKS displays on the status display panel in white:

```
TEXTFIELD(01,25,57,WHITE,NORMAL)
TEXTTEXT(DATA CENTER NETWORKS)
```

**Example**
In this example, all the function key settings will be displayed on line 24 of the Status Display panel:

```
TF(24,01,48,TURQUOISE,NORMAL)
TEXTTEXT(PF1=HELP 2=DETAIL 3=END 6=ROLL 7=UP 8=DN)
TF(24,51,79,T,N)
TT(10=LF 11=RT 12=TOP)
```

Notice that two entries are coded for the same line number.
Defining Function Keys on the Status Panel (PFKnn)

**Purpose**
The PFKnn entry defines all of the function keys that are unique to this panel.

**Syntax**

```
PFK
PFFn((command var)
```

**Parameters**

- **nn**  
  Values can range from 1 to 24.

- **command**  
  The command is issued when the defined function key is pressed.

- **var**  
  Variable can have the following values for this command:

  - **&COMP**  
    Identifies the component.

  - **&COMPAPPL**  
    Identifies the generic component within parentheses (for example, PU01(PU)). The generic component is shown only if the component was added using one.

  - **&ROOT**  
    Identifies the root or system.

  - **&SYSDATE**  
    Identifies the system date.

  - **&SYSTIME**  
    Identifies the system time.

  - **&IN or &INFO**  
    Identifies the detail entry word text that is displayed on the status panel.

  - **&DATE**  
    Identifies the date that the detail entry was added.

  - **&TIME**  
    Identifies the time that the detail entry was added.

  - **&SENDERID**  
    Identifies the reporter who submitted the detail entry.

  - **&SNODE or &SENDERNODE**  
    Identifies the node of the reporter who submitted the detail entry.

  - **&DA or &DATA**  
    Identifies the actual message text.

  - **&RV or &REFVALUE**  
    Identifies the reference value of the detail entry.

  - **&PR or &PRIORITY**  
    Identifies the priority of the detail entry.
&CO or &COLOR
Identifies to the color of the detail entry.

&HI or &HIGHLITE
Identifies the highlight level of the detail entry.

Usage
You can redefine all of the function keys except F3.

The definitions are active only when the status panel is displayed. The default settings defined with the initialization PFKnn statement are overridden.

F3 is always RETURN.

Example
In this example, the DIS PU01 command is issued when F4 is pressed with the cursor placed on the PU01 entry:
PFK4(DIS &INFO)
Defining the End of a Panel (ENDPANEL)

Purpose
The ENDPANEL entry identifies the end of a panel.

Syntax
ENDPANEL
Including Additional Members %INCLUDE

Purpose
The %INCLUDE entry includes additional members that contain panel definitions statements.

Syntax
%INCLUDE

Parameters
member
Use this keyword to specify the name of a member that defines a panel using the panel statements described in this section. The member must reside on a data set referenced by DSIPARM.
Chapter 4. Implementing Dynamic Display Facility (DDF)

This chapter describes the implementation of Dynamic Display Facility (DDF) and shows samples. You can use these samples to customize the DDF displays for your environment. These samples:

- Display network status on multiple panels
- Display network status on a single-panel
- Define a MessageView display
- Implement DDF in a focal-point environment
- Use operator MARK panels
- Store a DDF update under multiple components
- Group resources in DDF

Each sample in this chapter shows how DDF panels display when the sample is implemented, how DDF processes resource status, how you define the sample, and how to use it.

Displaying Network Status on Multiple Panels

When AON is shipped, a multiple panel display is sent as the default. This is the most useful display implementation. Figure 14 on page 72 shows a multiple panel display with numbered panels. The panels are described in the following section.
The Data Center Network panel is the first DDF panel. In Figure 7 on page 17, SNA and TCP/IP represent the SNA, and TCP/IP resources in the domain CNM01. To see the types of SNA resources defined to CNM01, move the cursor to SNA and press the down key, F8.

All the monitored SNA resource types within network CNM01 are displayed on the CNM01 Network Status panel. To see the names of the physical units that require operator attention defined in CNM01, move the cursor to PUS and press the down key, F8.

The next panel shows the names of the physical units that are experiencing problems. DDF shows each monitored resource in the color of its status. For example, PU001 is shown in red if the status is INACTIVE and requires operator intervention for recovery. To see the detail record, move the cursor to PU001 and press the detail key, F2.

The status details record for PU001 is shown on the Detail Status Display. The physical unit, PU001, requires operator intervention because a critical threshold has been reached and automation has stopped.
Multiple programs can update DDF at the same time with resource status without concern for the sequence or priorities posted by other programs. As AON automation resolves the problems, the resources are removed from DDF.

You can use the DDF MARK function to show that a problem is being addressed. To mark a problem, display the Detail Status Display for the problem and press F2. The operator ID is appended to the message at the bottom of the detail panel. When other users view DDF, their panel shows that someone is working on that particular problem. The MARK function eliminates the possibility of duplicated effort. The MARK and UNMARK commands are explained in "Using Operator MARK Panels" on page 86.

Understanding the Multiple Panel Display Function

The AON member EZLTREE contains entries for specific VTAM resource types, including NCP, LINES, and CDRMs when AON/SNA is installed. When a network problem occurs, AON adds a problem descriptor to DDF with the appropriate VTAM resource type. These problems propagate up the hierarchy that is defined in the EZLTREE member. AON arranges the problems in EZLTREE in order of severity. The highest level component reflects the most severe problem.

DDF displays network problems in a series of panels. AON provides a high-level system panel, an intermediate resource type panel, and a unique panel for each resource type. The high-level panel (EZLPNLST) shows the status of the problem with the highest priority in the system. The intermediate panel (FKVPNSNA) shows the color of the status of problem with the highest priority for unique resource types. Finally, the unique resource type panels show all resource names for a given resource type in order of the priority of the problem.

Defining Multiple Panel Displays

The following SNA-specific example requires these CNMPNL1 and DSIPARM members:
- EZLTREE
- EZLPNLST
- FKVPNSNA
- FKVPNLNA
- FKVPNLN
- FKVPNLC
- FKVPNLID
- FKVPNLL
- FKVPNLM
- FKVPNLN
- FKVPNLR

Chapter 4. Implementing Dynamic Display Facility (DDF)
The following sections describe how to customize the samples.

**Updating the EZLTREE Member**

Update the EZLTREE tree root name CNM01 to the name that is coded in your ENVIRON SETUP SYSNAME parameter. Refer to the following:

```c
/* NETWORK : CNM01 */
1 CNM01
  2 SYSTEM
  3 GATEWAY
  3 GROUPS
    4 CALIF
      5 LA
      5 SANFRAN
      5 SANDIEGO
    4 NEWYORK
    4 ATLANTA
  3 OPID
    4 OPER1
  3 NETWORK
    4 RESOURCE
      4 SNA
      5 SA
        6 NCP
        6 LINE
        6 LINKSTA
        6 CDRM
        6 CDRSC
        6 PU
        6 LU
        6 SESSION
        6 APPL
        6 ERR
      5 APPN
        6 CP
```
Updating the EZLCFG01 Member
To update the EZLCFG01 member:

1. In the ENVIRON SETUP definition in EZLCFG01, customize the SYSNAME value to reflect your domain ID (recommended). The SYSNAME can be any 5-character name used to identify this NETWORK domain. Refer to the *NetView for z/OS Administration Reference* for more information on tailoring the ENVIRON SETUP statement.

2. The DDF parameter in the ENVIRON DDF statement should be set to:

   DDF=STATUS if you want DDF to assign problem priority based on resource status. The DDF panel display shipped with AON is assigned by status.

Updating the EZLPNLS Member
To update the EZLPNLS member:

1. For AON/SNA, ensure %INCLUDE statements for the panels listed previously are present and not commented out.

2. Edit each of the panel samples and change all occurrences of CNM01 to the SYSNAME defined in the ENVIRON SETUP SYSNAME definition in EZLCFG01.

Updating the EZLINIT Member
Ensure the INITSCRN parameter is set to EZLPNLST.

Displaying Network Status on a Single Panel

In addition to viewing network status on multiple panels, you can view network status on a single panel.

Figure 15 on page 76 shows a single-panel display. The panels are described in the definition list that follows the figure.
The Data Center Network panel is the main panel from which you can display a number of unique networks. This is the first panel displayed when you invoke DDF. The NetView domains that send notifications to this DDF are CNM01, CNM02, and CNM03. For this example, suppose that CNM01 is highlighted in red, indicating an error status. The operator moves the cursor to CNM01 and presses F8 to move down in the hierarchy and display the CNM01 Network Status panel.

All resources requiring operator attention are color-coded by status. All resources in the network are displayed on this panel. You can customize the keys on this panel in EZLINIT "Defining Initialization Statements" on page 41.

The resource name that is displayed in the upper left corner has the highest priority of current problems. In this example, PU01 has the problem with the highest priority. PU01 is in red because the status is INACTIVE and requires operator intervention for recovery. The operator uses the Tab key to move the cursor to PU01 and presses F2 to display the Detail Status Display.

The Detail Status Display describes a problem for status descriptor PU01. PU01 requires operator intervention because a critical threshold was reached and automation stopped. To see only SNA resource failures (EZLPNL1), move the cursor to SNA and press F8.
Note that the Detail Status Display has function keys for the DDF commands, MARK (F2) and UNMARK (F10). With MARK, an operator can assign a status error to his ID from the Detail Status Display. The MARK and UNMARK commands are explained in "Using Operator MARK Panels" on page 86.

Understanding How Single-panel Displays Work

The AON member EZLTREE contains entries for specific VTAM resource types, including NCP, LINES, and CDRMS. In EZLTREE, all these VTAM resource types are defined one level lower than the AON component NETWORK. When a problem occurs in the network, AON adds a problem descriptor to DDF, using the appropriate VTAM resource type. When a problem is added using the resource type, it propagates up the tree so that the component NETWORK contains all network-related problems.

AON provides a single panel that shows, in priority order, all the resource names in the network that have problems. The panel uses the component name NETWORK to display all the network-related problems.

The priority of the problem is determined by what is coded on the ENVIRON DDF statement. If DDF=STATUS is coded, the VTAM status determines the priority of the problem. Resources that are inactive have the highest priority and are red. Resources with pending status have a lower priority and are pink or turquoise. Active resources are not displayed.

When DDF=TYPE is coded, the resource type that has the problem determines the priority. For example, if an NCP is inactive or pending, the problem has high priority and is red. If a PU has an error condition (inactive or pending), the problem has a lower priority and is yellow.

Defining Single Panel Displays

The values shipped in the samples are the default DDF values. The required members are:

- **EZLTREE**
  - Tree
- **EZLCFGDS**
  - Control File DDF status definitions
- **EZLPNLTY**
  - Network View Panel
- **EZLPNL1**
  - Resource View Panel
- **EZLPNLG1**
  - Gateway View Panel
- **EZLPNL03**
  - Message View Panel First
- **EZLPNL04**
  - Message View Panel Right
- **EZLPNL0C**
  - Message View Panel Down
- **EZLPNL0D**
  - Message View Panel Right
- **FKVPN1**
  - SNA Network View panel
**Updating the EZLTREE Member**

Update EZLTREE tree root name CNM01 to the SYSNAME defined in the ENVIRON SETUP statement of the control file. Refer to the following example:

```plaintext
/* NETWORK : CNM01 */
1 CNM01
  2 SYSTEM
  3 GATEWAY
  3 GROUPS
  4 CALIF
  5 LA
  5 SANFRAN
  5 SANDIEGO
  4 NEWYORK
  4 ATLANTA
  3 OPID
  4 OPER1
  3 NETWORK
  4 RESOURCE
  4 SNA
  5 SA
  6 NCP
  6 LINE
  6 LINKSTA
  6 CDRM
  6 CDRSC
  6 PU
  6 LU
  6 SESSION
  6 APPL
  6 ERR
  5 APPN
  6 CP
  6 EN
  5 X25
  6 X25MCH
  6 X25PU
```

**Updating the EZLCFG01 Member**

To update the EZLCFG01 member:

1. In the ENVIRON SETUP definition in EZLCFG01, customize the SYSNAME value to reflect your domain ID (recommended). The SYSNAME can be any 5-character name used to identify this NETWORK domain.

2. The DDF parameter in the ENVIRON DDF statement should be set to:
   - DDF=STATUS if you want DDF to assign problem priority based on resource status.
   - DDF=TYPE if you want DDF to assign problem priority based on resource type.

**Updating the EZLPNLS Member**

To update the EZLPNLS member:

1. For AON/SNA, ensure that the %INCLUDE statements for the panels listed previously are present and not commented out.

2. Edit each of the panel samples and change all occurrences of CNM01 to the SYSNAME defined in the ENVIRON SETUP SYSNAME definition in EZLCFG01.

**Updating the EZLINIT Member**

In the EZLINIT member, change the INITSCRN parameter to EZLPNLTY.
**Updating the EZLPNL1 Member**

/* DEFINE CNM01 NETWORK STATUS PANEL */
P(EZLPNL1,24,80,EZLPNLTY,EZLPNLTY,,)
TF(01,02,09,T,NORMAL)
TT(EZLPNL1)
TF(02,27,57,WHITE,NORMAL)
TT(CNM01 NETWORK STATUS)
SF(CNM01.NETWORK,04,05,16,N,,01)
ST( )
SF(CNM01.NETWORK,04,25,36,N,,02)
ST( )
SF(CNM01.NETWORK,04,45,56,N,,03)
ST( )
SF(CNM01.NETWORK,04,65,76,N,,04)
ST( )
...

SF(CNM01.NETWORK,20,25,36,N,,30)
ST( )
SF(CNM01.NETWORK,20,45,56,N,,31)
ST( )
SF(CNM01.NETWORK,20,65,76,N,,32)
ST( )
TF(24,01,52,T,NORMAL)
TT(PF1=HELP 2=DETAIL 3=END 4=DIS 5=CY 6=ROLL 7=UP 8=DN)
TF(24,53,79,T,NORMAL)
TT( 9=DEL 10=LF 11=RT 12=TOP)
EP

The tree shows NETWORK to be a level higher than any descriptors added using RESOURCE or under a resource type generic, for example CNM01.PU01(NU). The problem descriptors are propagated up to the status field of CNM01.NETWORK. All descriptors for the NETWORK are displayed on this panel.

If your network has more than 32 resources in an exception condition, duplicate this panel, customize the panel name, and add 32 to the descriptor count on each status field definition. This number causes descriptors 33 through 64 to display on this panel. Update the PANEL definition statement in EZLPNL1 to reflect a RIGHT panel with your new panel name (see "Defining the Panel Statements (EZLPNL1)" on page 57). Customize PANEL statement of your new panel to point LEFT to EZLPNL1.

---

**Defining a MessageView Display**

The following sample scenarios demonstrate MessageView. MessageView illustrates the use of status descriptor information types (refer to STATUSFIELD). Every detail descriptor is displayed on the panel along with its date, time, resource name, and message text. This panel provides a quick view of the DDF current descriptors. Figure 16 on page 80 shows a sample MessageView display.
The following example shows how the MessageView panel is defined.

```c
/* MESSAGEVIEW DISPLAY */
 *******************************************************************************/
P(EZLPNL01,24,80,EZLPNLST,EZLPNLST,EZLPNL02, ,EZLPNLOA)
TF(01,02,09,T,NORMAL)
TT(EZLPNL01)
TF(01,27,60,Y,NORMAL)
TT(AON: MessageView)
TF(03,22,29,T,N)
TT(Resource)
TF(03,34,45,T,N)
TT(Message text)
SF(CNM01.SYSTEM,04,02,09,N, ,D01)
ST( )
SF(CNM01.SYSTEM,04,12,19,N, ,T01)
ST( )
SF(CNM01.SYSTEM,04,22,29,N, ,C01)
ST( )
SF(CNM01.SYSTEM,04,32,79,N, ,M01)
:

TF(24,01,52,T,NORMAL)
TT(PF1=HELP 2=DETAIL 3=RET 6=ROLL 7=UP 8=DN)
TF(24,53,79,T,NORMAL)
TT( 9=DEL 11=RT 12=TOP)
PFK4()
PFK5()
EP
```

In this case, the D01 status descriptor locates the date of the first descriptor in the chain. T01 locates the time, C01 locates the component (usually resource name) and M01 locates the message text of the same descriptor.

---

**Figure 16. Message View Panel**

The following example shows how the MessageView panel is defined.
Implementing DDF in a Focal Point Environment

You can customize DDF to display the status of multiple networks at a focal point AON. You can implement notification forwarding from the distributed host to the focal point host.

In a focal point environment, DDF must be defined the same for each domain. The control file for each domain in a focal point environment should have either ENVIRON DDF,DDF=STATUS or ENVIRON DDF,DDF=TYPE if this code is defined on the focal points.

Viewing a Focal-point Environment

Figure 17 shows focal-point implementation.
For an explanation of each panel in Figure 17 on page 81, match the numeric callouts (for example, 1) beside the following paragraphs with the callouts beside each panel in Figure 17 on page 81.

1  The Data Center Networks panel is the main panel from which you can display a number of unique networks. This is the first panel that is displayed when DDF is invoked. The NetView domains sending notifications to this DDF are CNM01, CNM02, and CNM03. This example shows which panels are displayed when you select any of the domains from the Data Center Network panel.

2  All the monitored resource types within domain CNM01 are displayed on the CNM01 Network Status panel. This panel is tailored to display only resources in domain CNM01.

3  All the monitored resource types within domain CNM02 are displayed appear on the CNM02 Network Status panel. This panel is tailored to display only resources in domain CNM02.

4  All the monitored resource types within domain CNM03 are displayed on the CNM03 Network Status panel. This panel is tailored to display only resources in domain CNM03.

**Understanding How a Focal Point Environment Works**

AON forwards DDF updates to the focal point. At the focal point, define a tree structure for each domain in the EZLTREE member, so that DDF can determine how to store the update. This update works the same as it did on the originating domain.

To display the update, create unique panels for each domain. Use unique panel names and change the root name coded inside the panels to the root name specified on the originating system.

**Defining How a Focal-point Environment Works**

The values shipped in the sample in your DSIPARM data set contain the default DDF values.

**Updating the EZLTREE Member**

1. Copy EZLTREE to a new member, EZLTRE01.
2. Create new tree members for CNM02 and CNM03.
3. Copy EZLTRE01; change the member name and 01 root name.
4. Edit and change EZLTREE, so that EZLTREE contains only the includes for the new members. When completed, the members look like this:
Figure 18. EZLTREE Members
Updating the EZLPNLS Member

Create unique copies of existing panels for each domain. This requires some naming convention for the panels. In this example, the naming convention used is $xxxxP$, where $xxx$ is the domain ID. Depending on how you have chosen to display the network status, you must select one of the following sets of panels:

- AON/SNA multiple-panel display
- Single-panel display

For an AON/SNA multiple-panel display, select one of the following sets of panels:

- **EZLTREE**
  - Tree
- **EZLPNLST**
  - Main Panel
- **FKVPSNA**
  - Resource Type menu panel
- **FKVPNL**
  - APPLs menu panel
- **FKVPNLN**
  - NCP menu panel
- **FKVPNLN**
  - CDRM menu panel
- **FKVPNLD**
  - CDRSC menu panel
- **FKVPNL**
  - LINES menu panel
- **FKVPNL**
  - LINKS menu panel
- **FKVPNL**
  - PU menu panel
- **FKVPNL**
  - Resource (MISC—unrecognized resource type)
- **FKVPNL21**
  - All Resources menu panel
- **EZLPNLG**
  - Gateway menu panel
- **EZLPNL01**
  - MESSAGEVIEW menu panel First
- **EZLPNL02**
  - MESSAGEVIEW menu panel Down
- **EZLPNL0A**
  - MESSAGEVIEW menu panel Right
- **EZLPNL0B**
  - MESSAGEVIEW menu panel Down
- **FKVPNAC**
  - AON Control Point panel
- **FKVPNAE**
  - APPN End Nodes panel
- **FKVPA1**
  - All APPN Resources panel
- **FKVPNLX1**
  - X25 All Resources panel
- **FKVPNLX2**
  - X25 Machines panel
FKVPNXL3
X25 Inop SVCs panel

For single-panel display, select one of the following sets of panels:

**EZLTREE**
Tree

**EZLPNLTY**
Network View Panel

**EZLPNL1**
Resource View Panel

**EZLPNLG1**
Gateway View Panel

**EZLPNL03**
Message View Panel First

**EZLPNL04**
Message View Panel Right

**EZLPNL0C**
Message View Panel Down

**EZLPNL0D**
Message View Panel Right

**FKVPNL1**
SNA Resource View Panel

To update the EZLPNLS member:

1. Create unique panels for CNM02.
   Copy all panels for your display method, changing the panel name from EZLPNLxx to CNM02Pxx. For example, change EZLPNL1 to CNM02P1. Within each copied panel, change all occurrences of CNM01 to CNM02 and change all occurrences of EZLPNLxx to CNM02P (the EZLPNLTY and EZLPNLST—names should remain unchanged inside the panel). Commands assigned to DDF function keys may not be valid for resources in remote networks. Remove function key definitions that are not appropriate.

2. Create unique panels for CNM03.
   Copy all panels for your display method, changing the panel name from EZLPNLxx to CNM03Pxx. For example, change EZLPNL1 to CNM03P1. Within each copied panel, change all occurrences of CNM01 to CNM03, and change all occurrences of EZLPNLxx to CNM03P (except EZLPNLTY and EZLPNLST—these names should remain unchanged inside the panel). Commands assigned to DDF function keys may not be valid for resources in remote networks. Remove any function key definitions that are not appropriate.

3. Add pointers from the Main Data Centers panels to the new panels.
   For multiple panel display, add the following to EZLPNLST:
   ```
   ST(CNM02.NETWORK,04,09,18,N, ,CNM02P2)
   SF(CNM02)
   ST(CNM03.NETWORK,06,09,18,N, ,CNM03P2)
   SF(CNM03)
   ```

   For single-panel display, add the following to EZLPNLTY:
   ```
   ST(CNM02.NETWORK,04,09,18,N, ,CNM02P1)
   SF(CNM02)
   ST(CNM03.NETWORK,06,09,18,N, ,CNM03P1)
   SF(CNM03)
   ```

   When DDF is started, the CNM01 network status is primed. As the gateways to CNM02 and CNM03 are established, the DDF status for those networks are
forwarded to CNM01. If CNM02 or CNM03 are lost, the descriptors are deleted for that network and a COMMLOST entry is added to indicate that the network status is no longer current.

Using Operator MARK Panels

AON provides two commands for tagging each DDF detail panels with an operator ID. These commands are MARK (F2) and UNMARK (F10). With these commands, an operator can select network problems from DDF for analysis. Figure 19 illustrates operator MARK panels.

Figure 19. Operator MARK Panels (Part 1 of 2)
For an explanation of each panel in Figure 19, which starts on page 86, match the numeric callouts beside the panels (for example, 1) with the following:

To mark a problem, display the Detail Status Display for the problem and then press F2. This action appends the operator’s ID to the message at the bottom of the detail panel shown in 4b.

Marking a DDF problem also causes DDF to display the problem in reverse video, both on the Detail Status Display and on the upper level panels, as shown in 1 and 2b.

If problems are being handled by an operator, they are marked. Other operators can choose unmarked problems for analysis; thereby, preventing duplicate effort.

**Understanding Operator MARK Panels**

Because the status is displayed in reverse video, operators know at a glance the most severe problems are being addressed. In addition, because the problem is time-ordered, operators know which problems have been outstanding the longest.
From these DDF panels, operators use function keys to recycle the resource to which the cursor points, display those resources, mark descriptors, unmark descriptors, and delete descriptors. NetView commands work from these command lines.

---

**Defining Operator MARK Panels**

To define operator mark panels update the EZLTREE member used for your DDF with the operator ID appropriate for your DDF installation. To add operator IDs for the DDF MARK function, update EZLTREE:

```plaintext
/* NETWORK : CNM01 */
1 CNM01
  2 SYSTEM
  3 GATEWAY
  3 GROUPS
    4 CALIF
    5 LA
    5 SANFRAN
    5 SANDIEGO
    4 NEWYORK
    4 ATLANTA
  3 OPID
    4 OPER1
    4 OPER2
    4 OPER3
  3 NETWORK
    4 RESOURCE
    4 SNA
    5 SA
    6 NCP
    6 LINE
    6 LINKSTA
    6 CDRM
    6 CDRSC
    6 PU
    6 LU
    6 SESSION
    6 APPL
    6 ERR
    5 APPN
    6 CP
    6 EN
    5 X25
    6 X25MCH
    6 X25PU
```

When DDF deletes descriptors at the NETWORK level, it also deletes them from the panel of the operator. Define every operator ID, who will have independent panels, in the EZLTREE member.

---

**Grouping Resources in DDF**

Group resources in DDF by site, application, customer base, company, or by any other criterion. Figure 20 on page 89 shows a sample DDF group.
To view the resources in this DDF group, move the cursor to GROUPS and press Enter. Figure 21 shows the resources defined to GROUPS. This sample DDF group is by geographical area.

Figure 20. DDF Panel Showing GROUPS

To view the resources in this DDF group, move the cursor to GROUPS and press Enter. Figure 21 shows the resources defined to GROUPS. This sample DDF group is by geographical area.

Figure 21. Sample DDF Group Panel

Grouping Resources for Specific Requirements

The DDFGROUP control file entry groups DDF resources for display. For example, you can group dissimilar DDF resources by geographic location.

Using DDFGROUP statements, AON supports wildcards in DDFGROUP lists and places resources in multiple groups. During initialization, the EZLEAC11 program creates DDFRES control file entries from the DDFGROUP control file entries. To determine which groups are defined for a given resource, check only the DDFRES
control file entries. DDF resources are updated under both the applicable
DDFGENERIC and DDFGROUP matches.

The syntax of the DDFGROUP control file entry is:

```
DDFGROUP groupname,
  LIST=(res1,res2,...,resn)
  LIST=(res1,res2,...,resn)
  LIST=(res1,res2,...,resn)
  LIST=(res1,res2,...,resn)
```

`groupname`

The name of this group of DDF resources.

**Note:** You must use `groupname` in EZLTREE and the panel used to display
members of this group. Multiple DDFGROUP statements for the same
`groupname` are valid.

`LIST`

The list of resources in the group. Wildcards are supported in resource names.
You can create multiple lists. Each complete definition list must be on a single
line.

Refer to the [Tivoli NetView for z/OS Administration Reference](http://www.ibm.com) for information about
DDFGROUP syntax.

The EZLEAC11 program creates the DDFRES control file entries during startup in
this format:

```
DDFRES res1, GROUP=(group_name1, group_name2,...,group_nameN)
```

**Note:** AON creates all DDFRES control file entries.

The following sample list illustrates the format of DDFGROUP control file entries
that create a group of resources according to city:

```
DDFGROUP LA, LIST=(SALA*, GWATLA, GWLANY, GWFLA, GWSDLA)
DDFGROUP SANDIEGO, LIST=(SASD*, GWATSD, GWSDNY, GWFSFSD, GWSDLA)
DDFGROUP SANFRAN, LIST=(SASF*, GWATSF, GWFSNY, GWFSFSD, GWFSFLA)
DDFGROUP NEW YORK, LIST=(SANY*, GWATNY, GWFSNY, GWDSNY, GWLANY),
DDFGROUP NEW YORK, LIST=(IMNY1, IMNY2, IMSNY3)
DDFGROUP ATLANTA, LIST=(SAAT*, GWATNY, GWATSF, GWATSD, GWATLA),
  LIST=(CICSAT1, CICSAT2)
```

Some of the DDFRES control file entries that the EZLEAC11 program creates from
the above example are:

```
DDFRES SALA*, GROUP=(LA)
DDFRES GWATLA, GROUP=(LA, ATLANTA)
DDFRES GWLANY, GROUP=(LA, NEW YORK)
DDFRES GWSDLA, GROUP=(LA, SANDIEGO)
DDFRES GWFLA, GROUP=(LA, SANDIEGO)
```

In this example, SALA* is defined in group LA and SALACICS is defined in
SANDIEGO. Therefore, when SALACICS is added to DDF, it will be displayed
under the SANDIEGO group only, not under the LA group. However, if
SALACICS is explicitly defined in LA, then SALACICS is added to DDF under
both SANDIEGO and LA.

In this example, the Gateway nodes (GWsa1sa2) are defined in both of the cities
they are connecting. Each city has a subarea associated with it by naming
convention; therefore defining SAsa_id* is sufficient to assign all resources in the
city to the city status panel. As a result, gateways are displayed in several city
panels as shown in Figure 22. NEW YORK and ATLANTA are also responsible for associated applications shown in Figure 22.

![Figure 22. NEW YORK and ATLANTA Groups](image)

### Defining DDF Groups

The following sample definitions are based on the DDFGROUP control file entries shown in Figure 22. These definitions are located in DSIPARM.

Before you create DDF groups, consider your network and how you want the DDF panels to display. Refer to the Tivoli NetView for z/OS Administration Reference to code the DDFGROUP control file entries.

### Updating the EZLTREE DSIPARM Member

To update EZLTREE, add DSIPARM member `group_name` to the EZLTREE member. Use a level 3 element that is not subordinate to NETWORK, so the same message is not displayed in MESSAGEVIEW multiple times (one for each resource type and DDF group). The following example tree is valid:
Updating DDF Menus

To merge the new DDF panels into existing DDF panels, update the main panel menu, EZLPNLST, where you see CNM01, MESSAGEVIEW, and GATEWAY.

EZLPNLST is shown in Figure 24 on page 93.
EZLPNLST displays the Network Group Status Panel shown in Figure 25:

To view the CALIFORNIA subgroups, use the EZLPNLCA indicator panel shown in Figure 26 on page 94.

Figure 24. EZLPNLST Panel with DDF Group

EZLPNLGR displays the Network Group Status Panel shown in Figure 25:

Figure 25. EZLPNLGR Panel Showing a List of Cities

To view the CALIFORNIA subgroups, use the EZLPNLCA indicator panel shown in Figure 26 on page 94.
Creating Generic Display Panels for DDF Groups

Creating a generic panel for each new group, enables operators to see resources saved under them. The following is a generic panel:

```
/*******************************************************************************
/* DEFINE CNM01 NETWORK STATUS PANEL */
/* NEW YORK STATUS DISPLAY */
/*******************************************************************************
/* APAR# DATE DESCRIPTION */
/* -------- -------- --------------------------------------------- */
/*******************************************************************************
P(EZLPNLNY,24,80,EZLPNL2,EZLPNL2, ,)
TF(01,02,09,T,NORMAL)
TT(EZLPNLNY)
TF(02,21,59,WHITE,NORMAL)
TT(CNM01 NEW YORK CITY NETWORK STATUS)
TF(01,65,77,BLUE,NORMAL)
TT(PAGE 1 OF 1)
SF(CNM01.NEWYORK,04,05,16,N, , ,01)
ST( )
SF(CNM01.NEWYORK,04,25,36,N, , ,02)
ST( )
SF(CNM01.NEWYORK,04,45,56,N, , ,03)
ST( )
...
TF(24,01,52,T,NORMAL)
TT(PF1=HELP 2=DETAIL 3=END 6=ROLL 7=UP 8=DN)
TF(24,53,79,T,NORMAL)
TT( 9=DEL 10=LF 11=RT 12=TOP)
EP
/*******************************************************************************

Figure 26. EZLPNLCA Panel for CALIFORNIA Group

EZLPNLCA
CALIFORNIA NETWORK STATUS SUMMARY

LOS ANGELES
SAN FRANCISCO
SAN DIEGO

ALL CALIFORNIA RESOURCES

06/24/00 16:21:00
PF1=HELP 2=DETAIL 3=END 6=ROLL 7=UP 8=DN 10=LF 11=RT 12=TOP

94 Automated Operations Network Customization Guide
The generic group panels for ATLANTA, SANDIEGO, LA, and SANFRAN would look just like this one for CALIFORNIA except all occurrences of NEWYORK would be changed to the correct group name and the name of the panel would be updated.

**Updating the EZLPNLST CNMPNL1 Member**

To improve DDF performance, include new panels created for DDF groups in the EZLPNLST member of the CNMPNL1 data set for preloading. The DDF group sample panels provided with AON in the CNMPNL1 data set are:

- EZLPNLAT
- EZLPNLCA
- EZLPNLCI
- EZLPNLGR
- EZLPNLLA
- EZLPNLNY
- EZLPNLSF
Chapter 5. Issuing Dynamic Display Facility (DDF) Commands

AON routines use these commands to update the DDF status information from within DDF. If additional status information is required, use the following commands to update the DDF status information:

**DDF**    Displays DDF or a particular DDF panel
**DDFADD** Adds a status descriptor to a component
**DDFCLEAR** Clears DDF
**DDFDEL** Deletes a status descriptor from DDF
**DDFPANEL** Loads a panel member
**DDFQRY** Queries the status of specific status descriptors
**DDFTREE** Loads a tree member
**MARK** Marks a DDF resource with an operator ID
**UNMARK** Removes an operator ID from a DDF resource
Using the Dynamic Display Facility (DDF)

**Purpose**

The DDF command shows a color-coded status panel for the resources that are currently being acted upon by AON or require operator intervention for recovery. You can also use the DDF command to display a specific DDF panel.

**Syntax**

```
DDF
```

```
DDF (panel_name)
```

**Parameters**

`panel_name`

The name of the panel displayed in the upper left corner of the screen. Use this parameter to view a specific DDF panel.

**Usage**

- The workstation from which you enter the DDF command must support 3x79 Terminal Extended Attributes, which are blue, red, pink, green, turquoise, yellow, and white for color and blink, reverse video, and underscore for highlighting.
- If you enter DDF without the `panel_name` parameter, the main DDF panel is displayed. (The panel name for this panel is EZLPNLST.)
- The DDF command operates in full-screen mode only.

**Example**

To display the panel named FKVPNSNA when AON/SNA is installed, type the following:

```
DDF FKVPNSNA
```
Adding Status Descriptors (DDFADD)

Purpose

The DDFADD command issues a request to the EZLTDDF task to add a status descriptor to a status component with data supplied by a calling program, a command list, or NetView command facility (NCCF) operator. The date and time are provided automatically.

Syntax

```
DDFADD root_comp.stat_comp.(def_comp).keyword=value
```

Note: You can specify more than one `keyword=value` pairs.

Parameters

- `root_comp`
  The root component name that is defined in the root node of the tree structure. The root component is required because different systems can have status components with the same name defined in their respective tree structures. The root component must be unique, therefore, each status component in a tree structure can be uniquely identified by prefixing it with the root component entry.

- `stat_comp`
  A valid resource name or resource type, as defined in the EZLTREE member, which can contain up to 8 characters.

- `def_comp`
  The element in the DDF tree that organized DDF descriptors into groups. This element is usually a resource type, service point name, or group name defined in DDFGROUP control file entries. For example, you can define CNM01.LINE99(LINE) where CNM01 is the `root_comp`, LINE99 is the `stat_comp`, and LINE is the `def_comp`.

- `keyword=value`
  The status information that is necessary to add a status descriptor. In general, the only keywords that are needed are reference value, priority, and data. Valid values are:

  - `RefValue=value`
    Specifies reference value, which must be alphanumeric, with a maximum length of 20 (for example RV=NCP01).

  - `Info=text`
    Specifies the information displayed on DDF status panel in the STATUSTEXT field with a maximum length of 80 alphanumeric characters. The text must be delimited. Delimiters may be any unique character not appearing in the text. The length of the text field must be within the range specified by the start and end positions in the corresponding DDFADD command.
DDFADD

STATUSFIELD entry. See "Locating the Status Component (STATUSFIELD)" on page 60 for more information.

PRriority= value
Specifies a valid number within the range specified in the EZLINIT file. If the field is not specified, a program default priority of 999 is used.

COLOR = {Red | Blue | Turquoise | Green | Pink | Yellow | White}
Specifies color. If this is not specified, color is based on priority (recommended).

HighL = {Normal | Blink | Reverse | Underscore | Panel}
If this is not specified, highlighting is determined by the panel definition.

PropUp = {Yes | No}
Specifies propagation upward. If this is not specified, propagation up is governed by the entry in the EZLINIT file (recommended).

PropDwn = {Yes | No}
Specifies propagation downward. If this is not specified, propagation down is governed by entry in the EZLINIT file (recommended).

ProplvLU = {*|root_comp.stat_comp}
An asterisk (*) specifies that the level to which the status is to be propagated is the root of the tree. If a status component is specified, the status is only propagated up to, and including, the specified status component.

ProplvLD = {*|root_comp.stat_comp}
An asterisk (*) specifies that the status is to be propagated to the leaves of the tree. If a status component is specified, the status is only propagated down the leaves until, and including, the specified status component.

DataType = {OTH | MSG}
MSG specifies that message data is to be displayed on the status descriptor. OTH specifies that the data field contains user defined data.

Data = text
Specifies that text can be user or message data with a maximum length of 240 characters. Data may be composed of one or more lines. Each line of data can be delimited to produce a formatted display. Delimiters can be any unique character not appearing in the text. Delimiters between lines must be doubled. For example:

DA=#LINE1-DATA #### LINE2-DATA #### LINE3-DATA #

Causes the following to display on the status descriptor data field:

LINE1-DATA
LINE2-DATA
LINE3-DATA

Usage
If the data supplied in the DDFADD command exactly matches an existing descriptor in the chain, the new descriptor is not added.

With AON, status changes are logged in the automation log. The AON common logging routines provide the data for the DDFADD command and issue the DDFADD command. Any resource that is monitored by AON is automatically updated. User applications can also call the common logging routines to add DDF descriptors (recommended). The logging routines refer to the control file for status

Example

```plaintext
DDFADD CNM01.NCP001(NCP),RV=NCP001,PR=550,DT=MSG,
    DA='EZL509I NCP001 IS UNAVAILABLE (REPORTED BY AUTNET1)'
```

In this example, a status descriptor is added for NCP001 on CNM01 with a priority of 550. NPC001 is green on the status panel because priority 550 is in the green color range.
Clearing DDF (DDFCLEAR)

**Purpose**

The DDFCLEAR command deletes DDF elements based on resource. DDFCLEAR is a synonym for EZLEADCL.

**Syntax**

```
DDFCLEAR
```

**Parameters**

**STATUS**

Deletes records based on VTAM status. If you issue DDFCLEAR with no parameters, the default command issued is:

```
DDFCLEAR STATUS current_domain_id
```

**TYPE**

Deletes records based on resource type. TYPE = DDF records are deleted based on the VTAM resource type of the resource in the DDFname (usually in the format `domainid.resname(restype_generic)`). The resource is displayed with a D NET command. If the resource can not be displayed, the DDF element is not deleted. If the resource type matches a DDF type that has REQ=NOADD defined for it, the DDF element is deleted. Wildcards in the ENVIRON DDF statements are evaluated.

For example, AON default DDF statements provide for type of LU as REQ=NOADD. If DDFCLEAR were executed and some LUs were found among the DDF elements that now had an active status (because of activating the PU,SCOPE=U), the LUs would be deleted from DDF.

**AGE**

Deletes elements from DDF based on the age of the DDF element. A date and...
time stamp are stored into DDF when an element is added to DDF. This date and time are compared to the current date and time. If the difference exceeds the amount of DAYS or HOURS specified, the record is deleted. If you specify AGE with no other parameters, then the following command is issued:

```
DDFCLEAR AGE 5 DAYS current_domain_id
```

domain_id

The NetView domain ID to be evaluated for extraneous elements. The domain ID must match one of the level 1 values in EZLTREE. For a single domain DDF, this is usually the current domain ID. Check the value of the control file entry ENVIRON SETUP SYSNAME to determine this value. For a focal-point DDF representing several NetView domains, this value can be the EZLTREE root name or the ENVIRON SETUP SYSNAME value set up on any of the distributed systems. If no domain ID is provided, the current NetView domain ID is used.

clearamt

The number of days or hours to be used for an age threshold for deleting DDF records in long-lived NetView systems. This number must be a whole number. If you are deleting records by hours of age, the number is in the range of 0 — 24. If you are deleting by days, any number is valid up to 999. Because all DDF elements are kept in storage, they do not survive a recycle of NetView. For this reason, the AGE option would only be useful in environments where NetView has continuous availability for more than a week. If you issue DDFCLEAR AGE with no other parameters, the default value used is 5 days.

clearunits

Either DAYS or HOURS. All records whose DDF date is older than clearamt DAYS or clearamt HOURS is deleted. If no value is specified, DAYS is the default. If you specify HOURS, also specify a clearamt value because the parameters are positional.

Usage

Refer to the Tivoli NetView for z/OS Administration Reference for more information about the ENVIRON DDF control file entries.

Example

To clear all DDF records based on status, issue:

```
DDFCLEAR
```
Deleting Status Descriptors (DDFDEL)

Purpose

The DDFDEL command deletes status descriptors. Each status descriptor in the chain is compared against the data specified on the DDFDEL command. If each specified element in the command matches the data, the status descriptor is removed from the chain. The parameters supplied with the DDFDEL command can include wildcard characters. An asterisk (*) takes the place of a single character or, if used at the end of the field, indicates that the rest of the string should be automatically matched. If a keyword is not specified on the command, it is considered to be matched by any other value.

Syntax

```
DDFDEL
```

Note: You can specify more than one keyword=value pairs.

Parameters

**root_comp**

The root component name that is defined in the root node of the tree structure. The root component is required because different systems can have status components with the same name defined in their respective tree structures. Because the root component must always be unique, each status component in a tree structure can be uniquely identified by using a prefix with the root component entry.

**stat_comp**

Valid resource name or resource type that is defined in the EZLTREE member up to 8 characters.

**def_comp**

The element in the DDF tree that organized DDF descriptors into groups. This element is usually a resource type, service point name, or group name defined in DDFGROUP control file entries. For example, you can define CNM01.LINE99(LINE) where CNM01 is the root_comp, LINE99 is the stat_comp, and LINE is the def_comp.

**keyword=value**

Describes the status information that is necessary to delete a status descriptor. Valid values are:

- **PropDwn = [Yes | No]**
  Specifies propagation downward. If this is not specified, propagation down is governed by entry in the EZLINIT file (recommended).

- **RefValue = value**
  Specifies a reference value, which must be alphanumeric, with a maximum length of 20 characters (for example RV=NCP001).
INfo= text
Displays the information in the STATUSTEXT field, with a maximum length of 80 alphanumeric characters. The text must be delimited. Delimiters may be any unique characters not appearing in the text.

PRiority= value
Specifies the priority, which must be a valid number within the range specified in the EZLINIT PRI entry.

COLOR = {Red | Blue | Turquoise | Green | Pink | Yellow | White}
Specifies a color.

HIGHL = {Normal | Blink | Reverse | Underscore | Panel}
Specifies highlighting. If highlighting is specified in the DDFDEL command, it must be the same as that of the existing status descriptors, otherwise they are not deleted. It is recommended to let highlight default, because it is considered to be matched by any other value.

SOURCE= value
Specifies the operator under whom the DDFADD request was issued. This is the REPORTER field on the Detail Status Display panel.

DATA = {OTH | MSG}
MSG specifies that message data in the status descriptor is in message format. OTH specifies that the data field contains user defined data.

DATA= text
Specifies user or message data with a maximum length of 240 characters.

Usage
Use a wildcard character (*) with a single DDFDEL command to delete multiple status descriptors.

Example
In this example, any status descriptor for NCP001 on CNM01 with a reference value of NCP001 is deleted. Because no other keywords are used, no other criteria are used to determine a match:

DDFDEL CNM01.NCP001,RV=NCP001

Example
In this example, any status descriptors for NCP001 on CNM01 that have a reference value in the range of 100–190 are deleted:

DDFDEL CNM01.NCP001,RV=1*0

Example
In this example, any status descriptors for NCP001 on CNM01 that have a reference value beginning with R and a priority beginning with the number 3 are deleted:

DDFDEL CNM01.NCP001,RV=R*,PR=3*
Loading a Panel Member (DDFPANEL)

Purpose
The DDFPANEL command dynamically loads a panel member from the DSIPARM data set or deletes a panel member.

Syntax

```
DDFPANEL

(ADD | DELETE)
```

Parameters

`panel`
The name of the member that contains the panel to be loaded. The panel name and the member name must match.

Usage
When DDF is started, only the EZLPNLS panel definitions are loaded. Panels loaded with the DDFPANEL command are not reloaded during EZLTDDF initialization. If you add panels dynamically, restart DDF, add the panels again. If you update a panel and want the updated version to be displayed by DDF, use the DDFPANEL command to replace the current copy of the panel.

Example
The following command loads member NEWPANEL into memory, which gives you access to the panel defined in this member:

```
DDFPANEL NEWPANEL, ADD
```
Querying Status Descriptors (DDFQRY)

Purpose

The DDFQRY command queries the status of specific status descriptors. Parameters specified with the DDFQRY command can include wildcard characters. Asterisks (*) take the place of a single character or, if used at the end of the field, indicate that the rest of the string should be automatically matched. If a keyword is not provided on the command, any value is considered a match.

Syntax

```
DDFQRY
```

```
root_comp.stat_comp.(def_comp).keyword=value
```

Note: You can specify more than one `keyword=value` pairs.

Parameters

- **root_comp**
  The root component name that is defined in the root node of the tree structure. The root component is required because different systems can have status components with the same name defined in their respective tree structures. Because the root component must be unique, each status component in a tree structure can be uniquely identified by using a prefix with the root component entry.

- **stat_comp**
  Valid resource name or resource type (`def_comp value`) that is defined in the EZLTREE member. You can use up to 8 characters.

- **def_comp**
  The element in the DDF tree that organized DDF descriptors into groups. This element is usually a resource type, service point name, or group name that is defined in DDFGROUP control file entries. For example, define CNM01.LINE99(LINE) where CNM01 is the `root_comp`, LINE99 is the `stat_comp`, and LINE is the `def_comp`.

- **keyword=value**
  Describes the status information that can be used to query status descriptors. Valid values are:

  - **RefValue = value**
    Specifies reference value, which must be alphanumeric, with a maximum length of 20 characters (for example, RV=NCP001).

  - **INfo = text**
    Specifies information text, which must be alphanumeric, with a maximum length of 80 characters. The text must be delimited. Delimiters may be any unique character not appearing in the text (for example, IN='TAPE001').
DDFQRY

PRIority = value
Specifies the priority, which must be a valid number within the range specified in the EZLINIT entry PRIORITY (for example, PR=330).

CColor = {Red | Blue | Turquoise | Green | Pink | Yellow | White}
Specifies color. Status descriptors with the specified color are displayed if all other keyword values match.

HIghL = {Normal | Blink | Reverse | Underscore | Panel}
Specifies highlighting. Status descriptors with the specified highlighting are displayed if all other keyword values match.

SOurce = value
Specifies the operator ID that issued the DDFADD command. The ID can be a maximum length of 8 alphanumeric characters. This value is displayed in the status descriptor field.

DatATYPE = {OTH | MSG}
Status descriptors with the specified data type value are displayed if all other keyword values match.

DAta = text
Status descriptors with the same data text are displayed if all other keyword values match.

Usage
Use a wildcard character with a single DDFQRY command to view multiple status descriptors. AON uses DDFQRY to query DDF status information so that it can be forwarded from a distributed domain to a focal-point domain.

Example
In this example, status descriptors for NCP001 on CNM01 with a reference value of NCP001 are displayed:
DDFQRY CNM01.NCP001,RV=NCP001

Because no other keywords are used, an automatic match is assumed.

Example
In this example, status descriptors for NCP001 on CNM01 that have a reference values in the range of 100 — 190 are displayed:
DDFQRY CNM01.NCP001,RV=1*0

Example
Display all status descriptors saved with a def_comp value of NCP:
DDFQRY CNM01.NCP

Example
Display all status descriptors for the system CNM01:
DDFQRY CNM01
Loading Tree Members (DDFTREE)

**Purpose**

The DDFTREE command dynamically loads a tree member from the DSIPARM data set or deletes a tree member from storage.

**Syntax**

```
DDFTREE
```

```
ADD
DELETE
```

**Parameters**

`tree`

The name of the member that contains the tree structure to be loaded

**Usage**

The tree member can refer to other tree members with the `%INCLUDE` keyword. When a new tree is loaded to replace the existing tree, status descriptors that have similar `leaf` names in both trees are copied to the new tree.

When DDF is started, only the EZLPNLS panel definitions are loaded. Panels loaded with the DDFPANEL command are not reloaded during EZLTDDF initialization. If you add panels dynamically and restart DDF, add the panels again.

The DDFTREE command adds trees at the 01 or root level. Trees added exist with other trees in the EZLTREE member, unless the tree added has the same root as one that already exists. If the tree being added already exists in DDF, the current tree is replaced by the new one. You can replace only complete trees.

The member name of the tree and root of the tree must be the same for the DDFTREE command to replace a tree.

**Example**

The following command loads member NEWTREE into memory, which gives you access to the tree structure defined in this member:

```
DDFTREE NEWTREE,ADD
```
Assigning a Problem to an Operator in DDF (MARK)

Purpose

The MARK command assigns a DDF entry to an operator. Specify the data used to identify the DDF entry and the ID of the operator to be assigned to the entry.

Syntax

```MARK—root_comp.rv(opid)```

Parameters

- `root_comp`:
  The root component name that is defined in the DDF entry to be assigned.

- `rv`:
  The resource name that is displayed in the RefValue field of the DDF entry to be assigned.

- `opid`:
  The ID of the operator to whom this entry is to be assigned.

Usage

The `root` and `rv` parameters are required to issue this command from a command line. If the `opid` parameter is not specified, DDF assigns the entry to the operator ID issuing the MARK command.

If the entry to be signed out is already signed out no action is taken. Before attempting to reassign the entry to another operator, issue the UNMARK command. Once assigned, a resource remains assigned during recovery monitoring, as long as the operator who marked the resource is logged on. If the operator assigned to the resource logs off, the next recovery monitoring timer will cause the resource to be unassigned.

Example

To assign the DDF entry for NCP001 under the domain CNM01 to the operator OPER1, issue:

```
MARK CNM01.NCP001(OPER1)
```
Removing an Operator Assignment in DDF (UNMARK)

Purpose
The UNMARK command removes a DDF assignment entry from the specified operator. The data used to identify the DDF entry and the ID of the operator to be removed must be specified.

Syntax

```
UNMARK root_comp rv(operator_id)
```

Parameters

- `root_comp`  
  The root component name that is defined in the DDF entry to be unmarked
- `rv`  
  The resource name that is displayed in the RefValue field of the DDF entry to be unmarked
- `operator_id`  
  The operator ID assigned to this DDF entry

Usage

- The `root` and `rv` parameters are required. If you do not specify the `operator_id` parameter, the UNMARK command uses your operator ID as the default.
- If you issue the UNMARK command for a DDF entry that is not assigned to an operator, the UNMARK command takes no action.
- Each automation function has a syntax for specifying the name of the resource to be displayed (RV).

Example

To remove the operator ID, OPER1, from the DDF entry NCP001 under CNM01:

```
UNMARK CNM01.NCP001(OPER1)
```
Chapter 6. Creating Customized Procedures

Before you begin to write automation procedures for your network, become familiar with writing and interpreting NetView command list programs. Refer to Tivoli NetView for z/OS Customization: Using REXX and the NetView Command List Language for more information.

AON automation functions enable you to build and expand network automation. AON automation functions provide selected recovery routines and the ability to expand the recovery using common routines. To extend the automation capability functions of AON automation you can write programs or extend control file definitions. You can write extensions to:

- Check the status of network components using VTAM commands before taking action.
- Perform variable substitution for commands and replies with information obtained from the incoming message or from additional solicited information.
- Check automation settings for a resource, resource type, or network before taking action.
- Check specific error codes and take corrective action depending on the error code.
- Coordinate multiple independent messages into a single automated response.
- Create specialized operator dialogs.
- Check thresholds for events in a specified interval.

User-written extensions should interface with the common routines in AON when possible. Common routines provide the following benefits:

- Less code has to be written, reducing the development time
- Portability of the code if all installation unique information is maintained in the control file
- Consistent network management interface

Also note the following functions and features:

- Messages with HDRMTYPE='Y' are sent directly to the MVS console and cannot use the automation by the program operator interface (POI) to NetView. To circumvent this, use the SSI interface.
- Unsolicited VTAM messages sent to the operating system console (such as IST931I) are also sent to NetView by the POI if the PPOLOG start option is in effect. Message automation occurs when messages enter the NetView address space through the SSI task or through an operator task.
- You must define a subsystem interface (SSI) address space for the NetView machine on which AON is running. This enables AON to submit jobs for log maintenance, and to respond to console messages other than those from NetView. When you define the SSI, check your message processing facility list (MPFLST) in PARMLIB and ensure that all messages you want to automate can pass to and from NetView.
How Programs Use AON Control File Routines

The common routines enable a user-written program the capability to interface to the control file, status file, and log file with a minimum amount of coding. Functions, such as determining whether automation is available from the control file, are supported through the common routines.

To invoke a program that performs automated functions, use one of the following methods:
- Issue the program from the NetView automation table.
- Have the operator enter the program name or a synonym for the program.
- Have a higher level program call a lower level program.
- Issue a program with a timer.
- Issue a program with the NetView EXCMD command.

Invoking a program from the NetView automation table is commonly used for passive monitoring.

How a program is invoked and what function the program performs determines which AON routines the program calls. This section contains information that is applicable to all programs written to use the AON common routines, but the format and description is most applicable to programs which are invoked from the NetView automation table. The basic AON program:
- Performs program initialization processing.
- Checks recovery automation.
- Performs program unique processing such as THRESHOLDING and MONIT.
- Notifies operators of intervention required.
- Logs automation activity.

See Figure 27 on page 117 for the basic AON program flow.
Performing Program Initialization Processing

Initialization processing:
- Identifies the program with an &IDENT=name variable.
- Declares the common global variables (CGLOBALs) and task common global variables (TGLOBALs) that will be used.
- Issues debugging messages, if AONTRACE is turned on.
- Validates the program was properly invoked.
- Forces the program to be run under a specific operator.
- Saves the NetView message parameters in case the program uses the NetView WAIT statement.

These tasks are not required in user-written programs. However, to perform complex processing logic or to simplify the coding process, you might need to perform several of these housekeeping routines.

The identify step simplifies the coding of the message logging and notification routines. Declaring task global variables (TGLOBALs) and common global variables (CGLOBALs) is necessary if you need to use any of them. The AONTRACE function is useful in problem determination. Validating that the command was properly issued is useful to stop the operator from accidentally running the program. The validation can also be accomplished through NetView.
command authorization. If you use the NetView WAIT statement and still want to access the original message text or control information, save the variables.

See the coding in “Sample AON Extended Module” on page 119, the supplied extended programs, or the Tivoli NetView for z/OS Customization: Using REXX and the NetView Command List Language for further information about coding the initialization sections.

**Determining If Automation Is Available**

This step should always be performed by programs which are invoked from the automation table. The step is accomplished by calling the EZLECAUT common routine. The routine checks the RECOVERY policy definitions to make sure the automation flags enable automation. These checks eliminate the risk of automating messages for resources that should not be automated or for which automation is turned off.

**Performing Program Unique Logic**

The program unique logic may be any combination of AON common routines and user-written code. The requirements to be met and the functions to be performed determine the structure and content of the program. Several possible common routines and user-written code sections are:

- Activating resources
- Starting resource monitoring with or without reactivation attempts
- Logging messages and/or sending notifications
- Checking threshold values
- Interfacing with the file manager
- Writing user code for:
  - Issuing inquiries and checking answers
  - Setting common global variables (CGLOBALs) and task global variables (TGLOBALs) to control processing
  - Setting timer delays for reinitiating processing

**Activating resources**

You can activate VTAM resources by issuing an AON common routine, EZLEVACT.

The routine is necessary when an INACTIVE message is received and a user-written program is handling the message. VTAM commands or NetView commands can be used to assist in this.

**Recovery and reminder monitoring**

You can use recovery and reminder monitoring by issuing an EZLECATV or EZLESRMD AON common routine. These routines use the MONIT policy definitions. The EZLECATV routine starts monitoring the resource status and, if necessary, tries to recover the resource. The EZLESRMD routine monitors the resource and notifies operators, but does not start recovery.

**Updating status, DDF, logs, or sending notifications**

You can update status, DDF, logs, or send notifications by invoking the EZLEASLN common routine. The EZLEASLN common routine also generates alerts, based on your NOTIFY policy definitions.

This routine logs a message to the automation log (NLOG), DDF, and the NetView log (NETLOG). If you specify STATUS=YES, the status file is updated with the current automation status and acting operator. If you specify NOTIFY=YES on the EZLEASLN invocation, an operator notification is sent to all notification operators.
Checking thresholds
You can check thresholds, defined with the THRESHOLD policy definition, by invoking an EZLEATHR AON common routine.

This routine is used if it is necessary to track and maintain a threshold count. A threshold is used to change the recovery action based upon the threshold level that was exceeded. The common routine also records the current date and time as a new error to the status file.

Interfacing with the file manager
You can interface with the file manager by using the file manager interface commands.

The commands are used if you need to access uniquely defined entries in the control file or want to directly inquire or update the status file.

Several file manager interface commands exist. See “Chapter 7. Using AON Command Processors” on page 123 for more information.

User code for issuing inquiries and checking answers
You can write user code to issue inquiries and check answers by using NetView or MVS commands to check the value or status of some non-VTAM owned resource, system component, or subsystem before continuing.

Refer to the WAIT command in “Tivoli NetView for z/OS Customization: Using REXX and the NetView Command List Language” for more information.

User code for setting CGLOBALs and TGLOBALs to control processing
You can write user code for setting CGLOBALs and TGLOBALs to control processing by using NetView commands to set a flag indicating progress, message counts, and other various indicators that must be kept from one occurrence of a message or event to the next.

Refer to the discussion of CGLOBALs and TGLOBALs in “Tivoli NetView for z/OS Customization: Using REXX and the NetView Command List Language” for more information.

User code for setting timer delays for reinitiating processing
You can write user code for setting timer delays to reinitiate processing by using NetView commands.

Use this code when a program must reissue itself or another program after a given time period to perform additional processing. One example is in the active monitoring of failed network resources.

Refer to the discussion of AT, AFTER, and DELAY commands in the “Tivoli NetView for z/OS Command Reference” for more information.

Sample AON Extended Module
This section describes the AON extended module that tries to recover the resource name that is passed to it. The module uses a subset of the AON common routines. The notes following the example refer to portions of the module and the AON common routines that are described in “How Programs Use AON Control File Routines” on page 116.

********************************************************************
/*
/* (C) Copyright IBM Corp 1997, 2001. All rights reserved.
/*
/*********************************************************************************/
/* DESCRIPTION: THIS CLIST WILL ATTEMPT TO RE-ACTIVATE THE FAIL RESOURCE. */
/* WHERE CALLED: INVOKED FROM THE MESSAGE TABLE BY USER CODED MESSAGE TABLE ENTRY SENDING RESOURCE NAME ONLY */
/* ITS PURPOSE IS TO REACTIVATE THE RESOURCE AUTOMATICALLY. IT ALSO INFORMS NCF ALERT OPERATORS IF THE RESOURCE CAN NOT BE ACTIVATED. IT WILL ALSO CHECK THRESHOLDS SET BY THE CUSTOMER TO INFORM THE ALERT OPERATORS OR TERMINATE RECOVERY. */

/*---------------------------------------------------------------*/
trace off
Parse source . invok ident .
pars upper arg ArgString
Parms = ArgString
/*---------------------------------------------------------------*/
/* Entry Trace Call */
/*---------------------------------------------------------------*/
/SF5800001/SF590000
'GLOBALV GETC EZLTRACED EZLTRACEO.'Opid()||, /* Trace Variable */
' NETOPER NETOPER2 BASEOPER'
If Substr(EZLTRACED,1,4) <> 'NONE' Then /* If not NONE ... */
Do
/SF5800002/SF590000
'EZLTRACE ENTRY 'ident argstring /* Entry trace */
Trace Value(EZLTRACER)
End
OperTrace = Substr(Value('EZLTRACEO.'Opid()),1,2)
Return_Code = 0
/* If opid is valid then process, else route to BASEOPER autotask */
if opid() = Netoper2 | opid() = Netoper | opid() = Baseoper then
Call PROCESS
Else /* route recovery */
'Do /* route recovery */
  /* EZLEASCD AON Baseoper EZLEAGEN 'Parms */
/SF5800003/SF590000
'AONCMD 'Baseoper' EZLEAGEN 'Parms
Return_Code = 9
End /* route recovery */
Call Leave_Now

PROCESS:
Resname = Parms
/4 'EZLEAGRN 'Resname
'GLOBALV GETT RESTYPE RESTSTAT '
/* Issue EZL509I - resource UNAVAILABLE message */
/5 'EZLEASLN NOTIFY=Y,STATUS=Y,'|| |
  'AON,INACTV,EZLEAGEN,EZL509,'date("U")','|| |
  substr(time(),1,5),'Resname','Restype','domain()
Call ChkAuto
Return
/*---------------------------------------------------------------*/
/* PERFORM AUTOMATION */
/*---------------------------------------------------------------*/

CHKAUTO:
/* Call the Check_Auto routine to check the RECOVERY Policy */
/6 'EZLECAUT' Resname Restype
if Rc ≠ 0 then /* don't auto resource? */
  Call EXIT0
  'EZLEACKT' Resname /* Ck for exist timer */
if Rc = 1 then /* If timer exists? */
  Call EXIT1
if substr(Resstat,1,3) = 'ACT' then /* If resource is ACT */
  Call EXIT0
if substr(Resstat,1,3) = 'CON' then /* If resource is CONCT*/
  Call EXIT0
/* Call Thresholding routine to check the INFREQ/FREQ/CRIT Thresholds*/
  'EZLEATHR' Resname Resname Resname OPTION=SNA'
if Rc = 0 then /* thresh ok? */
  Call RECOVERY /* attempt recovery */
else
  'VARY NET,ID='Resname',INACT,F' /* inactivate it */
Return

/* ATTEMPT RECOVERY OF THE RESOURCE. */
RECOVERY:
  'EZLECATV' Resname Resname Resname OPTION=SNA'
  Call EXIT0
EXIT0:
  Return_Code = 0
  Call Leave_Now
EXIT1:
  Return_Code = 1
  Call Leave_Now

The common global variables and task global variables are defined.
EZLTRACE is called to perform AON standard entry tracing.
This section of code ensures the module is running under an automated
operator, and if not, sends the command to the NETOPER or BASEOPER
automation operator to run.
Common routine EZLEAGRN is called to get resource information.
An EZL509I message is logged regarding the availability of the resource.
The common routine EZLECAUT is called to verify that automation is
enabled. The resource type and status will be set from this module. If the
recovery flag is on, the recovery module runs.
The existence of a timer for this resource is checked. If a timer exists with
timer ID equal to the resource name, recovery is already in effect for this
resource through another means. If recovery is already in effect, the
program (command list) exits.
The resource is analyzed to determine whether a critical threshold has been
exceeded. If it has, the resource is forced into the inactive state and the
module exits.
EZLECATV is called to initiate recovery for the resource and continue recovery monitoring based on MONIT intervals.
Chapter 7. Using AON Command Processors

AON provides command processors to interface with the control file, status file, or automation log. These command processors update and retrieve data contained in the control file, status file, or automation log. AON provides the following command processors:

- “Control File Interface Command (EZLCFG)” on page 124
- “Log File Interface Command (EZLLOG)” on page 130
- “Status File Interface Command (EZLSTS)” on page 133
- “Using the SNBU Automation Status File Interface (FKVSNBU)” on page 137
Control File Interface Command (EZLCFG)

Purpose

Use the EZLCFG command to load, display, and modify the control file entries. You can also use the command to display the status of the control file function or to perform a syntax check on the control file member. Before you can display panels and modifications, you must load the control file into storage. Once loaded, the panels and modifications affect an in-storage version of the control file, providing a temporary update facility. To permanently change the control file member, edit the DSIPARM data set, and reload.

Note: There are certain restrictions when you update the control file. Command updates to entries in the control file cannot be performed online if the control file entry is longer than 200 characters. This is a NetView restriction. To code an entry that is longer than 200 characters, change the configuration file and reload it.

The control file entries are keyword oriented and, therefore, independent of the NetView command list language or the operating system. Modifications or displays using the EZLCFG command capitalize upon the use of these keywords to retrieve entries for individual resources, resource types (such as all NTFYOPS), or system wide defaults. This keyword structure decreases the need to generate command list variables that contain information for each resource.

The EZLCFG command functions both as an operator command and a command list interface to the control file.

Note: Use of the EZLCFG command is discouraged. You should use the POLICY command. The EZLCFG command is still available for migration purposes. For additional information on the POLICY command refer to the [Tivoli NetView for z/OS Automation Guide](#).

Syntax

**EZLCFG**

```
MEMBER=member

STATUS

REQ=
  DEL
  REPL

ENTRY=entry

TYPE=
type
(typ)

SEARCH=
  (parm1=val1...,parm5=val5)
  keyword=val
```

Note: You can specify more than one `keyword=value` pairs.
Parameters

**MEMBER**

Specifies the member name containing the control file entries. This member will be loaded into storage for subsequent use. The member must exist in a data set that is concatenated with the data sets in the DSIPARM DD statement.

Using EZLCFG MEMBER=member might cause problems since all policy file(s) to be loaded are defined in CNMSTYLE and are loaded using the POLICY command.

To load or test the control file, enter EZLCFG MEMBER=member[,TEST].

**STATUS**

Displays the name of the logical policy file loaded, or a message stating that the control file function is inactive.

**ENTRY**

Specifies the entry field of the control file. The entry-name must be 1 to 32 characters, without embedded blanks, commas, or quotes.

For display, delete, replace, or add requests:

**TEST**

Enables the user to perform very simple syntax checking of a control file member. Entries are not verified.

*Note:* MEMBER is the only operand allowed with TEST.

**REQ**

Specifies the type of request. DISP (display) is the default request if not entered. DEL (delete) and REPL (replace) must be entered when performing that function. REPL will add an entry if one does not exist.

**TYPE**

The type field on the control file. The default TYPE field is an asterisk (*) which is allowed only for DISP (display) commands. The specification of asterisk (*) returns all the type fields associated with a given entry name. For example, all resource entries or all NTFYOP entries.

You can specify up to five types of DISP, DEL, and REPL requests when performing the command. Only the first type found in the list of types is affected by the command.

When specified, *type* must be 1 to 32 characters, without embedded blanks, commas, or quotes.

**SEARCH**

Valid only when used with REQ=DISP. Searches all entries specified by ENTRY and TYPE for a match where *parms=value* defines the data to be associated with the specified ENTRY and TYPE fields. The *parm* can be any character data, without embedded blanks, commas, or quotes. The character (=) and the *value* must immediately follow the *parm*. The *value* can have embedded quotes, commas, and blanks as long as single quotes or parentheses are around the *value*. The *parms=value* field has a restriction of 70 characters.

When multiple search arguments are specified, an implied logical ‘OR’ operation is performed. Entries that match one of the search arguments will be presented.
EZLCFG

AON defines a number of ENTRY, TYPE and parms=value fields. The JOB=jobname parameter in the resource control file entry is an example of the parms=value field.

Usage

The following wildcard characters can be used:

*     Multiple character wildcard
%     Single character wildcard

For example, both PU0* and PU%0 matches for PU01. SEARCH=(AUTO=*) matches for entries containing the AUTO parameter. ENTRY=ENVI* matches for all entries starting with ENVI.

When an EZLCFG REQ=DISP command is entered that requests a certain ENTRY with a one or more specific TYPES, it will search for those types in the order they were specified in the command. When the first match is found, the information is returned to the requestor as a multiline message. If there are no matches, it performs a final search with a type-name of DEFAULTS for that ENTRY. If there is still not a match, a message is returned to the requestor. If the type-name of DEFAULTS is found, that information is returned to the requestor.

When performing a display, if a specific TYPE is found, it is treated as a COMPLETE entry. Only that specific entry is displayed.

To replace an existing entry, type the entire new entry. The REPLACE command is treated as a COMPLETE entry. The existing entry and the replacement entry will not merge. This replacement process has a limitation on the length of the REPLACE command. The maximum length command is 240 characters. When you attempt to replace a control file entry, the replacement entry can be no greater than 200 characters. To do a large replace command, use the INPUT 3 NetView command to expand your command buffer.

KEYWORD=value operands in the control file can be no longer than a single line.

Note: The control file member must be placed in storage using the PIPE INSTORE command. Prior to placing the item in storage, the SUBSYM stage must be invoked to resolve system symbolics. Failure to perform these actions prior to the EZLCFG call will result in the stored version being used. Therefore, without invoking the SUBSYM stage, any changes to the control file on the DASD are ignored.

All policy definitions are loaded into NetView storage using PIPE INSTORE. If you make changes to your policy files in DSIPARM, you can use POLICY REQ=LOAD to reload you new policy definitions and remove the existing policy definitions.

Usage:

The POLICY command supports all functions that EZLCFG performs except the search function. You should use the POLICY command whenever possible.

Messages

The following messages will be issued during the successful operation of the EZLCFG command.
For the test function (TEST):
EZL026I TEST OF THE CONTROL FILE MEMBER "member-name" WAS UNSUCCESSFUL

For the status function (STATUS):
EZL005I MEMBER member-name CURRENTLY BEING USED FOR THE CONTROL FILE

For delete and replace commands:
EZL001I REQUEST "request" WAS SUCCESSFUL

For DISPLAY commands:
EZL111I AUTOMATION CONFIGURATION DISPLAY - ENTRY= entry-name
EZL112I ACTIVE TYPE= type-name , DESIRED TYPE= type-name1 ...
EZL113I DATA IS parms=value
EZL002I END

**Note:** Multiple EZL113I messages might be displayed after an EZL112I command and where the type-name is an asterisk (*). When the type is omitted or specified as an asterisk (*), the DESIRED TYPE is not displayed on the EZL112I message. For example, the following may occur:

EZL111I AUTOMATION CONFIGURATION DISPLAY - ENTRY= NTFYOP
EZL112I ACTIVE TYPE= NETOP1
EZL113I DATA IS OPER='OPER 1'
EZL113I DATA IS CLASS=(10,40)
EZL112I ACTIVE TYPE= NETOP2
EZL113I DATA IS CLASS=(10)
EZL002I END

The following messages will be issued if the entry or member name was not found or the request failed.

For load function (MEMBER=):
EZL042I MEMBER member-name NOT FOUND

For the test function (TEST):
EZL027I THE FOLLOWING ERRORS ENCOUNTERED IN PROCESSING MEMBER member-name
EZL023A FIELD "KEYWORD='VALUE('" CONTAINS UNBALANCED PARENTHESIS
EZL029I ENTRY TYPEA,KEYWORD='VALUE('
EZL026I member-name ERROR DISPLAY
EZL004I PROCESSING FAILED FOR "EZLCFG MEMBER=member-name" COMMAND
EZL026I TEST OF THE CONTROL FILE MEMBER "member-name" WAS UNSUCCESSFUL

For the status function (STATUS):
EZL040I CONTROL FILE INACTIVE

For the delete and DISPLAY commands:
EZL041I UNABLE TO FIND type name

**Note:** The REPLACE command will add an entry for a message if one does not exist.

Any messages beginning with EZL0 other than those previously documented should be considered an error situation.

**Example**

This example contains the command that displays the RECOVERY flag for PU01. The control file entry is:
The command is:
EZLCFG REQ=DISP,ENTRY=RECOVERY,TYPE=PU01

The response is:
EZL111I AUTOMATION CONFIGURATION DISPLAY - ENTRY= RECOVERY
EZL112I ACTIVE TYPE= PU01 , DESIRED TYPE= PU01
EZL113I DATA IS AUTO=Y
EZL113I DATA IS NOAUTO=(TUESDAY,10:00,12:00)
EZL002I END

In this example, a RECOVERY flag exists for the PU01 resource. The operator or command list processes the command to display the entry and the associated response is a multi-line message.

Example

This example shows a command to display the RECOVERY flag for PU0. The control file entries are:
RECOVERY DEFAULTS,AUTO=Y,NOAUTO=(MONDAY,10:00,12:00)
RECOVERY PU01,AUTO=Y,NOAUTO=(TUESDAY,10:00,12:00)

The command is:
EZLCFG REQ=DISP,ENTRY=RECOVERY,TYPE=PU01

The response is:
EZL111I AUTOMATION CONFIGURATION DISPLAY - ENTRY= RECOVERY
EZL112I ACTIVE TYPE= DEFAULTS , DESIRED TYPE= PU01
EZL113I DATA IS AUTO=Y
EZL113I DATA IS NOAUTO=(MONDAY,10:00,12:00)
EZL002I END

The previous example causes no change in the displayed data. Although a DEFAULTS flag exists, data from the different RECOVERY types (PU01 and DEFAULTS) will not merge.

Example

This example contains a command to display the RECOVERY flag for PU01. The control file entry is:
RECOVERY DEFAULTS,AUTO=Y,NOAUTO=(MONDAY,10:00,12:00)

The command is:
EZLCFG REQ=DISP,ENTRY=RECOVERY,TYPE=PU01

The response is:
EZL111I AUTOMATION CONFIGURATION DISPLAY - ENTRY= RECOVERY
EZL112I ACTIVE TYPE= DEFAULTS , DESIRED TYPE= PU01
EZL113I DATA IS AUTO=Y
EZL113I DATA IS NOAUTO=(MONDAY,10:00,12:00)
EZL002I END

The previous example shows the EZL112I message with the found type-name (ACTIVE) as DEFAULTS, but the requested type-name (DESIRED) is PU01. This occurs because there was no RECOVERYPU01 flag, therefore the EZLCFG command automatically searches for a type-name of DEFAULTS. For this example, a RECOVERY DEFAULTS entry existed; therefore that information was displayed.
Assuming a DEFAULTS entry did not exist, the command would have resulted in an EZL041I UNABLE TO FIND TYPES... message.

Example

This example contains a command to display the RECOVERY flag for PU01 or PU, depending on which one exists. The control file entry is:

RECOVERY PU, AUTO=Y

The command is:

EZLCFG REQ=DISP,ENTRY=RECOVERY,TYPE=(PU01, PU)

The response is:

EZL111I AUTOMATION CONFIGURATION DISPLAY = ENTRY= RECOVERY
EZL112I ACTIVE TYPE= PU , DESIRED TYPE= PU01
EZL113I DATA IS AUTO=Y
EZL002I END

The EZL112I message shows that the type-name found (ACTIVE) is PU, the requested types (DESIRED) were PU01 and PU. The EZLCFG command searched first for the RECOVERY PU01 flag; when none was found, the command searched for the RECOVERY PU flag.
Log File Interface Command (EZLLOG)

The EZLLOG function is used to update and view the automation log. The syntax of the EZLLOG function is:

**Syntax**

```plaintext
EZLLOG

EZLLOG

EZLLOG: ID=resource, FUNC=function, STATUS=status, OPID=operator-id

EZLLOG: DOMAIN=domain-id, FROM=list, DESC=message_text
```

**Parameters**

- **ID**: Resource associated with this log record
- **FUNC**: Feature that wrote this record. The function entries in the automation log indicate which function in network automation wrote the record. This field can be any four characters.
- **STATUS**: Automation status of the resource. For a list of the status codes, see "Status File Interface Command (EZLSTS)" on page 133.
- **OPID**: Requesting operator ID.
- **DOMAIN**: Originating domain ID
- **FROM**: Name of the command list or operator that generated the record to the log.
- **DESC**: Description text to be written in log record. The format is
  ```plaintext
  DESC=msgno text
  ```
  Where the variables are:
  - **msgno**: Number of the message that generated the log entry.
  - **text**: Free-form text describing the log event.

  The maximum length is 240 bytes.

**Usage**

The log records are:
- Availability records
- Information records
- Errors and debug records
Use availability records to create an availability trend report. AON writes records to the log when a resource becomes unavailable or when the resource becomes available again.

Information records describe the activities of the automation process. These are the AON tracking records.

Errors and debug records are written when an error has occurred (for example, wait time-out) or when the debugging feature is in effect.

The EZLLOG and EZLALOG commands, and the EZLTLOG task are:

**EZLLOG**

This command can be issued from the operator’s terminal, program, or another command processor. EZLLOG performs syntax checking necessary before sending the request to the automation log task (EZLTLOG). Such errors as valid length and required parameters are checked. If it detects errors, it returns a message explaining the error to the originator of the request.

**EZLALOG**

This command is run as a data services command. The EZLALOG command writes the record to the automation log file and returns messages to the requestor indicating the success or failure of the request.

**EZLTLOG**

This task handles the necessary interfaces to perform the NetView facilities. When EZLTLOG receives the internal function request (IFR) from EZLLOG, it issues the appropriate command, in this case EZLALOG.

To start the automation log task, the following command must be entered on the operator command line of a NetView screen or from the NetView initial startup program using STARTEZL:

```
STARTEZL LOG
```

The following messages display when the task has been initiated:

```
DSI166I EZLTLOG IS ACTIVATED BY operator_ID
DSI530I EZLTLOG : DST IS READY AND WAITING FOR WORK
```

**Example**

The command list writes a record to the automation log indicating that a resource has recovered. This example is for illustrative purposes and may not reflect current AON code:

```bash
'GLOBALV GETC DOMAINID'
ReqDomid = Domainid

EzIstatus = 'ACTIV'

Ezlmsgtxt = ez1emsg('EZL504','N',Clsnm,Resname,Restype)

if N1ogname = '' then
    N1ogname = SUBSTR(Resname,1,8)

if N1ogfunc = '' then
    if LENGTH(Ezltower) <= 4 then
        N1ogfunc = Ezltower
    else
        N1ogfunc = SUBSTR(Ezltower,1,4)

if pos('/',clsnm) > 0 then
```

Chapter 7. Using AON Command Processors 131
parse var clsnm clsnm '/'.

if length(clsnm) > 8 then
    clsnm = substr(clsnm,1,8)

'TRAP AND SUPPRESS MESSAGES ONLY EZL006I EZL009I EZL01* EZL020I ',
'EZL030I DSI002I CNM421I'
Logcmd = 'EZLLOG '||
        'ID='Nlogname',FUNC='Nlogfunc',OPID='Reqopid',DOMAIN='||,
        Reqdomid',FROM='Clsnm',STATUS='Ezlstatus',DESC='Ezlmsgtxt
Logcmd = LEFT(Logcmd,240)
Logcmd

The previous command results in the following entry in the automation log:
EZL504I HOST AVOSTIN IS AVAILABLE (REPORTED BY NTCOPUN6)

If you page to the left in the automation log, you see the following:
13:44:31 EZLEREVCV AUTMSG2 NV6K NORMAL AVOSTIN CNM01
Status File Interface Command (EZLSTS)

Purpose

Use the EZLSTS command to add, display, change, and delete the status file records. The records are maintained in a VSAM data set. The EZLSTS command interfaces with the VSAM file to maintain control information vital to AON. The critical information maintained is:

- Automation status
- Whether an error threshold has been exceeded
- Time and date information for error conditions. All entries use GMT timestamps.

The EZLSTS command functions as an operator command and as a command list interface to the status file.

Note: Use of EZLSTS as an operator command should be restricted, because this command can modify any field on the status file, and incorrect use can lead to unpredictable results.

Every field defined in the status file is updated by certain automation routines. User-written command lists should use common routines to perform the updates, to ensure proper updating of all the fields. Use the EZLSTS command only to display information.

Syntax

Use the following syntax for adding, updating, displaying, or deleting a single entry:

\[
\text{EZLSTS}
\]

\[
\text{EZLSTS} \text{-ID=resource} \left(\text{DISP}, \text{REQ=resource}, \text{TYPE=resource}, \text{DEL}, \text{REPL}\right) \left(\text{ID=resource}, \text{DISP}, \text{REQ=resource}, \text{TYPE=resource}, \text{DEL}, \text{REPL}\right)
\]

\[
\left(\text{OPID=operator_id}, \text{NOTIFY=\{Y\}N}, \text{STATUS=status}\right) \left(\text{OPID=operator_id}, \text{NOTIFY=\{Y\}N}, \text{STATUS=status}\right)
\]

\[
\left(\text{MOA\text{R}=\{monitor\text{ed-time}\}}, \text{DATE=error-date}, \text{TIME=error-time}\right) \left(\text{MOA\text{R}=\{monitor\text{ed-time}\}}, \text{DATE=error-date}, \text{TIME=error-time}\right)
\]

\[
\left(\text{THRESHLD=\{CRIT\}FREQ\{INFR\}}\right) \left(\text{THRESHLD=\{CRIT\}FREQ\{INFR\}}\right)
\]

Use the following for displaying multiple records:
EZLSTS

Parameters

ID=resource
Specifies the 16 character resource ID that is the key to the status file record. This ID is the resource name for resource records.

FROM=resource
Specifies the 16 character resource ID that is the starting key when displaying multiple status file records. This resource is the resource name for resource records.

TO=resource
Specifies the 16 character resource ID that is the ending key when displaying multiple status file records. If not specified, the value defaults to the same key as the FROM parameter. This resource is the resource name for resource records.

REQ={DISP|DEL|REPL}
Specifies the type of request. DISP (display) is the default request if not entered. DEL (delete) and REPL (replace) must be entered when performing that function. REPL will add a record if one does not already exist.

OPID=operator_id
The NetView operator ID which requested or is performing the status update. If not supplied, the EZLSTS command will use the current operator ID. This is only applicable when the STATUS parameter is used.

TYPE=resource
Specifies the 10-character resource type for display and summary purposes. This value is the resource type for the specific resource. For example, physical units are represented by PU and NCPs are represented by NCP. Other values can be defined, if desired.

STATUS=status
Specifies the resource status. The parameter can be any valid status value. The following is a list of valid status codes generated by AON automation:

- INACTV: Resource is inactive.
- ACTIVE: Resource is active.
- INRCVY: Resource is in recovery.
- RCVAUT: Resource recovered by AON automation.
- RCVSYS: Resource recovered by the system.
- TUSER: Resource recovery terminated by user.
- TTTHRS: Resource recovery terminated by automation thresholds.
- REACTV: Reactivation process enabled.
- TREACT: Recovery halted; reactivation intervals exceeded.
- MANUAL: Manual intervention is required.
- NCPDMP: NCP dump has a reply to (AON/SNA only).
- NCPRLD: NCP reload has a reply to (AON/SNA only).
- ERRORS: Command list error conditions.
- TRACKS: Command list tracking records.
- REMIND: Reminder processing enabled.
- REMTRM: Reminders halted; reminder intervals exceeded.
- IUSER: Resource deactivated by user.
- RECVRY: Resource currently being recovered by automation.
NOTIFY={Y|N}
   Specifies whether a notification message was sent to the operator for a
   specified condition.

MONITOR=monitor-Time
   Specifies the time a monitor or status change command was issued in HH:MM
   format. Time is in military time (00:00 to 23:59).

DATE=error-date
   Specifies the date of this error time stamp being recorded, in MM/DD/YY
   format. Date is used for threshold purposes.

TIME=error-time
   Specifies the time of this error time stamp being recorded, in HH:MM format.
   Time is used for threshold purposes.

THRSHLD={CRIT|INFR|FREQ}
   Specifies the threshold which has been exceeded. Used for operator monitoring
   of thresholds.

Usage
If a user-written command list performs a replace and update function, only those
fields that need replacing have to be specified. No change occurs to the other fields
on the status file. The EZLSTS command has a maximum length of 240 characters,
so a REPLACE/UPDATE command is limited to the 240 characters.

Messages
The following messages will be issued during the successful operation of the
EZLSTS command. For the DELETE and REPLACE commands, a message such as
the following is issued:
EZL001I REQUEST "request" WAS SUCCESSFUL FOR "resource"

For the DISPLAY commands, the following messages are issued for PU01:
EZL150I STATISTICS DISPLAY REQUESTED FOR PU01
EZL151I ID= PU01 , TYPE= PU , STATUS= ACTIV
EZL152I LAST UPDATE BY OPERATOR AUTNET2
EZL153I LAST THRESHOLD EXCEEDED = INFR
EZL155I OPERATOR NOTIFIED: Y , TIMERSET:
EZL156I LAST MONITORED DATE= 01/06/00 , TIME= 13:41
EZL157I LAST MONITORED DATE= 01/06/00 , TIME= 13:41
EZL160I ERROR COUNT DATE TIME
EZL161I 01 01/06/00 13:35
EZL161I 02 01/06/00 13:40
EZL002I END

Note: For DISPLAY commands, EZL160I and EZL161I messages that contain error
time stamp information can be replaced with EZL159I if error timestamp
information does not exist.

Messages are issued if the entry or member name is not found. For the DELETE
and DISPLAY commands, a message such as the following is issued:
EZL041I UNABLE TO FIND RECORD "resource"

Note: The REPLACE command will add an entry if one does not exist, so
successful message would result.

Any message beginning with EZL0, other than those previously documented, is an
error.
Example

A command that displays the status record for PU01 follows:

EZLSTS REQ=DISP,ID=PU01

The response follows:

EZL150I STATISTICS DISPLAY REQUESTED FOR PU01
EZL151I ID= PU01 , TYPE= PU , STATUS= ACTIV
EZL152I LAST UPDATE BY OPERATOR AUTMSG
EZL153I LAST THRESHOLD EXCEEDED - INFR
EZL155I OPERATOR NOTIFIED: Y , TIMERSET:
EZL156I LAST STATUS CHANGE DATE= 05/06/00 , TIME= 09:17, OPID= AUTNET2
EZL157I LAST MONITORED DATE= 05/06/00 , TIME= 09:17
EZL160I ERROR COUNT DATE TIME
EZL161I 01 05/06/00 14:21
EZL002I END

The operator or command list processes the command to display the entry, and the associated response is a multiline message.

Example

The following command adds or replaces the PU01 status value:

EZLSTS REQ=REPL,ID=PU01,STATUS=INACT

The response follows:

EZL001I REQUEST "REPL" WAS SUCCESSFUL FOR "PU01 "

Automated Operations Network Customization Guide
Using the SNBU Automation Status File Interface (FKVSNBU)

Purpose

The FKVSNBU command is a command processor that provides an interface to the SNBU automation status file. With the FKVSNBU command, you can add, display, change, or delete AON/SNA SNBU automation status file records from the AON status file.

The EZLSTS file tracks the status of the lines that are in switched speed or in backup. AON/SNA SNBU automation maintains these status file entries.

Syntax

Use the following syntax for single records:

```
FKVSNBU
```

```
FKVSNBU
  ID=SUpu_name,REQ=DISP
  STATUS=status,HIER1=ncp

  ,HIER2=line,HIER3=pu_name
  ,ALTPORT=alt_port
  ,POOL=pool

  ,SPEED=FULL
  ,LEVEL=2
```

Use the following syntax for multiple records:

```
FKVSNBU
```

```
FKVSNBU
  FROM=res_name,TO=res_name
  ,REQ=DISP
```

Parameters

**ID** Identifies the status file record. If the record is for an alternate port, the first two characters are SU followed by the PU name or the NCP line name. This naming convention prevents AON/SNA entries from changing the AON/SNA SNBU entries when PU errors occur.

**REQ** Specifies the type of request. You can use the following types:

- **DISP** Displays the default. If you use DISP, code the ID parameter only.
- **DEL** Deletes a status file record. If you use DEL, code the ID parameter only.
- **REPL** Replaces a status file record. If a record did not already exist, REPL adds a new one.
FKVSNBU

STATUS
Identifies the AON/SNA SNBU automation status. The status codes are:

INIT Identifies initial status for speed selection (operation begun).
ISNBU Initiates AON/SNA SNBU (operation begun).
BOTH Transmits and receives lines in backup speed.
LOCAL Transmits side of line in backup speed.
RCV Receives side of line in backup speed.
SNBU Indicates successful dial connection (operation completed).
BUSY Indicates alternate port is in use.
DOWN Indicates alternate port is inoperative.

HIER1 Identifies the first resource name in the hardware monitor alert hierarchy. For AON/SNA SNBU automation, it is the name of the NCP.

HIER2 Identifies the second resource name in the hardware monitor alert hierarchy. For AON/SNA SNBU automation, it is the name of the line.

HIER3 Identifies the third resource name in the hardware monitor alert hierarchy. For AON/SNA SNBU automation, it is the name of the PU.

ALTPORT Names the alternate port in use, if specified in the control file entry for the PU.

POOL Identifies the modem pool from which the current modem was chosen.

LEVEL Specifies the modem link segment level when in a tailed-circuit modem configuration. The default is 1.

SPEED Indicates whether the modem is operating at full speed or whether it has been switched to backup speed by AON/SNA SNBU automation.

FROM Used with the FKVSNBU command, displays the AON/SNA SNBU status file records beginning with those for this resource.

TO Used with the FKVSNBU command, displays the AON/SNA SNBU status file records up to those for this resource.

Usage
To display multiple status records, use the FROM and TO parameters. REQ=DISP is the default. No other parameters are necessary.

Example
The following is a sample EZLSTS record:
EZEI501 STATISTICS DISPLAY REQUESTED FOR SU uname
EZEI511 ID= SU uname , TYPE= SNBU , STATUS= SNBU
EZEI521 LAST UPDATE BY OPERATOR oper id , RECORD TYPE=
EZEI531 LAST THRESHOLD EXCEEDED -
EZEI551 OPERATOR NOTIFIED: N
EZEI561 LAST STATUS CHANGE DATE= 11/07/00 , TIME= 08:20 , OPER= oper id
EZEI571 LAST MONITORED DATE= 11/07/00 , TIME= 08:20
EZEI621 HIER 1= ncpname , 2= linename , 3= puname , 4= , 5=
EZEI631 ALTPORT= ********** , POOL= **** , SPEED= ********** , LEVEL= *
EZEI002 END
Chapter 8. Coding Common Routines

Common routines provide easy-to-use generic functions for expanding automation capabilities beyond those provided and supported by AON. These generic functions can be used to reduce development time when creating procedures or extending those provided.

Transferring information and checking the control file are examples of tasks where common routines are applicable. A user-written routine calls one of these routines from the message table, control file, or extended routine to perform a specific task.

AON also provides user exits that can be called by the common routines. User exits perform functions such as syntax or threshold checking.

Table 8 lists the common routines described in this section.

<table>
<thead>
<tr>
<th>Name</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>CGLOBAL</td>
<td>&quot;Using the Common Global Variable Command Processor (CGLOBAL)&quot; on page 143</td>
</tr>
<tr>
<td>EXIST</td>
<td>&quot;Querying Command Availability (EXIST)&quot; on page 143</td>
</tr>
<tr>
<td>EZLEATDF</td>
<td>&quot;Calculating Time (EZLEATDF)&quot; on page 143</td>
</tr>
<tr>
<td>EZLE1UFW</td>
<td>&quot;Forwarding User Messages (EZLE1UFW)&quot; on page 146</td>
</tr>
<tr>
<td>EZLEACKT</td>
<td>&quot;Checking the Timer (EZLEACKT)&quot; on page 147</td>
</tr>
<tr>
<td>EZLEAGEN</td>
<td>&quot;Recovering Generic Resources (EZLEAGEN)&quot; on page 148</td>
</tr>
<tr>
<td>EZLEAGRN</td>
<td>&quot;Getting Resource Information (EZLEAGRN)&quot; on page 149</td>
</tr>
<tr>
<td>EZLEASLN</td>
<td>&quot;Updating the Status File and Logging Messages (EZLEASLN)&quot; on page 151</td>
</tr>
<tr>
<td>EZLEATHR</td>
<td>&quot;Checking Thresholds (EZLEATHR)&quot; on page 152</td>
</tr>
<tr>
<td>EZLECALL</td>
<td>&quot;INFORM Action (EZLECALL)&quot; on page 160</td>
</tr>
<tr>
<td>EZLECATV</td>
<td>&quot;Using Active Monitoring and Recovery (EZLECATV)&quot; on page 161</td>
</tr>
<tr>
<td>EZLECAUT</td>
<td>&quot;Checking Automation (EZLECAUT)&quot; on page 163</td>
</tr>
<tr>
<td>EZLEFAIL</td>
<td>&quot;Processing Generic Failures (EZLEFAIL)&quot; on page 164</td>
</tr>
<tr>
<td>EZLEMCOL</td>
<td>&quot;Setting Panel Message Color (EZLEMCOL)&quot; on page 170</td>
</tr>
<tr>
<td>EZLEMSG</td>
<td>&quot;Formatting Panel Messages (EZLEMSG)&quot; on page 171</td>
</tr>
<tr>
<td>EZLEFRM</td>
<td>&quot;Driving the Inform Policy (EZLEFRM)&quot; on page 172</td>
</tr>
<tr>
<td>EZLENTFY</td>
<td>&quot;Notify Policy List (EZLENTFY)&quot; on page 173</td>
</tr>
<tr>
<td>EZLERAIP</td>
<td>&quot;Setting the AIP User Status Bit (EZLERAIP)&quot; on page 174</td>
</tr>
<tr>
<td>EZLERCMD</td>
<td>&quot;Routing Commands over Cross-Domain Sessions (EZLERCMD)&quot; on page 176</td>
</tr>
<tr>
<td>EZLERECV</td>
<td>&quot;Recovering Resources (EZLERECV)&quot; on page 177</td>
</tr>
<tr>
<td>EZLEGRWY</td>
<td>&quot;Routing Commands to Other NetView Domains (EZLEGRWY)&quot; on page 177</td>
</tr>
<tr>
<td>EZLEROUT</td>
<td>&quot;Routing NNT Cross-Domain Logon Information (EZLEROUT)&quot; on page 181</td>
</tr>
<tr>
<td>EZLERTVE</td>
<td>&quot;Retrieving AON Information (EZLERTVE)&quot; on page 183</td>
</tr>
<tr>
<td>EZLESRMD</td>
<td>&quot;Issuing Resource State Reminders (EZLESRMD)&quot; on page 183</td>
</tr>
<tr>
<td>EZLESTOP</td>
<td>&quot;Stopping Cross-domain Sessions (EZLESTOP)&quot; on page 186</td>
</tr>
</tbody>
</table>
Table 8. AON Common Routines (continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>EZLESTRT</td>
<td>“Starting Cross-domain Sessions (EZLESTRT)” on page 183</td>
</tr>
<tr>
<td>EZLEVACT</td>
<td>“Activating VTAM Resources (EZLEVACT)” on page 189</td>
</tr>
<tr>
<td>EZLEVINA</td>
<td>“Deactivating VTAM Resources (EZLEVINA)” on page 190</td>
</tr>
<tr>
<td>EZLEVMOV</td>
<td>“Moving VTAM Resources (EZLEVMOV)” on page 191</td>
</tr>
<tr>
<td>EZLSMSU</td>
<td>“Sending MSUs to an MS Transport Application (EZLSMSU)” on page 192</td>
</tr>
<tr>
<td>EZLTRACE</td>
<td>“Running Entry and Exit Traces (EZLTRACE)” on page 194</td>
</tr>
<tr>
<td>FKVESYNC</td>
<td>“SNA Resource Automation (FKVESYNC)” on page 202</td>
</tr>
<tr>
<td>FKXECNVT</td>
<td>“SNMP RFC Conversion (FKXECNVT)” on page 198</td>
</tr>
<tr>
<td>FXXETRA1</td>
<td>“Syntax of FXXETRA1 Program” on page 200</td>
</tr>
<tr>
<td>IPCMD</td>
<td>“TCP/IP Command Support (IPCMD)” on page 198</td>
</tr>
</tbody>
</table>
Using the Common Global Variable Command Processor (CGLOBAL)

Purpose

The CGLOBAL routine displays the names and value associated with common global variables. Running the CGLOBAL routine displays the message EZL016I containing the name of the common global variable and the message EZL017I containing the value associated with the common global variable.

Syntax

CGLOBAL

Parameters

\textit{varname}

The specific name of a global variable or a name pattern with wildcard characters (* or %)

Example

Issuing \texttt{CGLOBAL domainid} returns the following messages:

\begin{verbatim}
EZL016I NAME = DOMAINID
EZL017I VALUE = CNM01
EZL002I END
\end{verbatim}

The message EZL017I shows that CNM01 is the value associated with the common global variable domainid.

Example

Issuing \texttt{CGLOBAL \ast time} returns the following messages:

\begin{verbatim}
EZL016I NAME = RCMDTIME
EZL017I VALUE = 90
EZL016I NAME = XDOMTIME
EZL017I VALUE = 120
EZL016I NAME = WAITTIME
EZL017I VALUE = 29
EZL016I NAME = DIALTIME
EZL017I VALUE = 120
EZL002I END
\end{verbatim}

The EZL016I messages show all of the command global variables that end with the characters \texttt{time}. The EZL017I messages show the values associated with each command global variable.
Querying Command Availability (EXIST)

Purpose

EXIST determines whether the specified command is available and sets a return code as its response. Only libraries in the DSICLD or STEPLIB concatenation are searched. The command must be a valid command list, REXX program, or NetView command processor.

Syntax

```plaintext
EXIST
```

Parameters

- `command`
  
  Name of a command

Return Codes

- 0  Valid command
- 4  Operator is not authorized to issue the command
- 8  Error (incorrect command name or no command specified)
- 16 Not found

Example

The following REXX fragment is an example of how you can use EXIST:

```rexx
/* Check to see if BLOG command is installed, and call if found */
'EXIST BLOG'
ExistRC = RC
if ExistRC = 0 then
do
  'BLOG'
end
```
Calculating Time (EZLEATDF)

Purpose
The EZLEATDF routine calculates the difference between two date and time
stamps and returns the values in TGLOBALs in terms of days, hours, minutes, and
time. Keep the parameters for the EZLEATDF routine in the same order as they are
shown in the following syntax diagram.

Syntax

```
EZLEATDF
```

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>start_date</td>
<td>Earliest date (in mm/dd/yy format)</td>
</tr>
<tr>
<td>start_time</td>
<td>Earliest time (in hh:mm format)</td>
</tr>
<tr>
<td>end_date</td>
<td>Later date (in mm/dd/yy format)</td>
</tr>
<tr>
<td>end_time</td>
<td>Later time (in hh:mm format)</td>
</tr>
</tbody>
</table>

Return Codes

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Program processed successfully.</td>
</tr>
<tr>
<td>5</td>
<td>Parameters are not valid.</td>
</tr>
</tbody>
</table>

Task global variables

- **TIMEDIFF**: The number of hours and minutes between start and end (after the
difference in days is calculated)
- **DAYSDIFF**: The number of days between start date and end date
- **HOURDIFF**: Total hours and minutes between start date and end date (in hh:mm
  format)
- **MINDIFF**: Total minutes between start time and end time ((hh*60) + mm). To
  express the total difference in minutes, add the HOURDIFF global (converted to minutes)
to the MINDIFF value.

Example

This program call:
```
EZLEATDF 06/23/00 06:28 06/24/00 15:50
```

Sets the following task global variables:
```
TIMEDIFF  =  09:22
DAYSDIFF =  2
HOURDIFF =  57:22
MINDIFF  =  3442
```
Forwarding User Messages (EZLE1UFW)

**Purpose**

The EZLE1UFW routine forwards messages to an AON focal-point domain.

**Syntax**

```
EZLE1UFW
```

```
EZLE1UFW
    CLASS=class
    MSG='text'
```

**Parameters**

**CLASS=class**

Specifies message classes used to determine which notification operators receive this message. You can specify up to ten message classes. There are no default message classes. The AON notification classes are in the range 00—99. The different notification classes are described in [Tivoli NetView for z/OS Messages and Codes](https://www.ibm.com/itm/docs/nvmsgs/).

**MSG='text'**

Specifies the text used for this message. If not coded, the text in the message buffer is used. Although this variable is usually optional, it is required if not called from the automation table.

**Return Codes**

- 0: Program (command list) processed correctly.
- 1: Processing error was encountered.

**Usage**

The MSG parameter cannot be used for multiline messages.

**Example**

The following example sends individual messages for each line in the multiline response:

```lisp
IF MSGID='IST075I'
THEN EXEC(CMD('EZLE1UFW CLASS=20')ROUTE(ALL*));
```
Checking the Timer (EZLEACKT)

Purpose
The EZLEACKT routine determines whether a timer exists and stores pertinent timer data in task global variables (TGLOBALs).

Syntax
EZLEACKT

Parameters
		
timerid
Specifies the ID of the timer being searched for. Typically, for AON recovery efforts, this is the resource name.

Return Codes
0 Timer does not exist.
1 Timer exists.

Task global variables
EZLTIMCM Command to be issued under timer ID
EZLTIMOP Operator under which the timer would run
EZLTIMTM Time of day the timer would pop
EZLTIMDT Date the timer would pop

Usage
The EZLEACKT routine is used primarily to determine if automation is already in effect for a resource with an existing timer ID, such as timers from MONIT intervals.
Recovering Generic Resources (EZLEAGEN)

Purpose

EZLEAGEN is a generic module that can do the following:
- Attempt resource recovery
- Check error thresholds
- Check automation recovery flag
- Check resource status
- Initiate recovery on MONIT intervals

Syntax

```plaintext
EZLEAGEN
```

Parameters

resource

The resource to be recovered

Usage

This routine is provided as a model for user-written modules extending AON techniques. The EZLEAGEN module can be started by sending it a valid VTAM resource name.

Example

See “Sample AON Extended Module” on page 119 to see how EZLEAGEN is coded.
Getting Resource Information (EZLEAGRN)

Purpose
The EZLEAGRN routine retrieves VTAM information on a resource through a VTAM display command and stores the data in task global variables (TGLOBALs).

Syntax

```plaintext
EZLEAGRN
```

Parameters

- `res_name` The name of the VTAM resource to be displayed

Return Codes

- 0 Resource values assigned
- 1 Module error

Task global variables

Table 9 lists the EZLEAGRN routine TGLOBALs.

<table>
<thead>
<tr>
<th>TGLOBAL</th>
<th>Description</th>
<th>Retrieved from VTAM message</th>
</tr>
</thead>
<tbody>
<tr>
<td>RESTYPE</td>
<td>Resource type</td>
<td>IST075I</td>
</tr>
<tr>
<td>RESSTAT</td>
<td>Resource status</td>
<td>IST486I</td>
</tr>
<tr>
<td>RESMAJ</td>
<td>Resource's major node</td>
<td>IST134I or IST081I</td>
</tr>
<tr>
<td>RESLINE</td>
<td>Resource’s higher node-line</td>
<td>IST081I</td>
</tr>
<tr>
<td>RESNODE</td>
<td>Resource’s adjacent major node</td>
<td>IST391I</td>
</tr>
<tr>
<td>RESPU</td>
<td>Resource’s higher node controller</td>
<td>IST135I</td>
</tr>
<tr>
<td>RESSA</td>
<td>Resource’s subarea</td>
<td>IST484I</td>
</tr>
<tr>
<td>RESSW</td>
<td>Resource’s switched major node</td>
<td>IST136I</td>
</tr>
<tr>
<td>RESNET</td>
<td>Resource’s network ID</td>
<td>IST075I or IST1043I</td>
</tr>
</tbody>
</table>

Usage

VTAM resource types are translated into the following standard automation resource categories:
- **LU** Logical unit
- **PU** Physical unit
- **PU** LCL
- **LINE** Line
- **PU** PU*
All other resource types remain the same. If the resource cannot be displayed in VTAM, a resource type of DEFAULTS is issued and message EZL208I is logged.

The EZLEAGRN routine gathers information about SNA resources. However, your installation may require you to tailor the information gathered by the EZLEAGRN routine. You can code user exits to alter this information. You can update values for the EZLEAGRN routine with the EXIT05 parameter of the ENVIRON EXIT control file entry. Refer to the Tivoli NetView for z/OS Administration Reference for more information about the ENVIRON EXIT entry.
Purpose

The EZLEASLN routine updates the AON status file, logs associated messages in the AON log file and NetView log, and issues notifications. The content of the log or notification message depends on the message ID and the variable data that is placed in the message.

This routine can only be initiated by another module or by a command processor. Keep the parameters for the EZLEASLN routine in the same order that they are shown in the following syntax diagram. Even though you do not use some of the parameters, include the comma delimiters as if those parameters were in the routine.

Syntax

```
EZLEASLN
  NOTIFY= Y, N, Event_Type
  STATUS= Y, N
  ,function
  ,status, modulename, msgno, date, time, trig, res, msg_type, res_type
  ,parm
```

Parameters

NOTIFY
Specifying Y enables notifications to an operator and message forwarding of some event. Specifying Y queries the NOTIFY Policy based on ResName and ResType.

Specifying N does not notify anyone, it will just log the message to DSILOG.

Specifying Event_Type queries the NOTIFY Policy for the type of event such as:
- CRITTHRS
- REMIND
- NOMOMONS
- NAMESERV

STATUS
Specifying Y enables updates to the status file with the following data:
- RESSTAT
- RESTYPE
- OPID
function

Defines the AON component that wrote the message. The function can be:

- AON
- APPN
- IP390
- NVAIX
- SA
- SNA
- SNBU
- TCPIP
- X25

status

Specifies the resource status. This value updates the information in the status file and logs it in the AON log. The following is a list of some of the valid status codes generated by automation:

- ACTIVE  Resource is active.
- ERRORS  Command list error conditions.
- INACTV  Resource is inactive.
- INRCVY  Resource is in recovery.
- IUSER   Resource deactivated by user.
- MANUAL  Manual intervention is required.
- NCPDMP  NCP dump has been replied to (AON/SNA only).
- NCPRLD  NCP reload has been replied to (AON/SNA only).
- RCVAUT  Resource recovered by AON automation.
- RCVSYS  Resource recovered by the system.
- REACTV  Reactivation process enabled.
- RECVRY  Resource currently being recovered by automation.
- REMIND  Reminder processing enabled.
- REMTRM  Reminders halted; reminder intervals exceeded.
- TRACKS  Command list tracking records.
- TREAT  Recovery halted; reactivation intervals exceeded.
- TTTHRS  Resource recovery terminated by automation thresholds.
- TUSER   Resource recovery terminated by user.

modulename

Specifies the name of the module or program that is issuing the message. This parameter is used in the AON log, by the tracking and problem determination processes.

msgno

Specifies the number of the message to be issued.

date

The date the message was logged. Use the NetView variable, &DATE or REXX function date ('U').
time
The time the message was logged. Use the NetView variable, &TIME or REXX function TIME('U').

{trig | res}
Specifies the ID of the resource or the specific cause for the message being issued. For errors and tracking information, the message trigger typically is the command that caused the error message to be issued. The message trigger may also be the operator ID, a parameter keyword, a control file keyword, or other similar indicative information.

The trig variable can be a maximum of eight characters in length and should not contain spaces.

msg_type | res_type
Specifies the resource type or the general cause for the message being issued. For errors and tracking information, the message type is typically ERR or ERROR. The message type may also be INFO, INIT, or similar indicative values.

The msg_type variable can be a maximum of 10 characters in length.

parms
User parameters are message-dependent fields that fill in the message content. Up to 5 user parameters can be coded, but a restriction of 240 characters in the EZLEASLN call may reduce the actual number of parameters that can be coded.

Return Codes
0 Message request processed successfully.
4 An error has occurred. Browse the log for further details.
9 Message request has been routed to a MSGOPER autotask.

Usage
The EZLEASLN routine can perform the following functions:
- Message logging
- Status update and message logging
- AON notification processing based on the NOTIFY policy

Use the EZLEASLN routine when logging messages. Messages produced by calling this routine are not displayed at operator stations if NOTIFY=N is specified.

It can also be used to display a message to the operator. Messages produced by calling this routine are displayed at a NetView terminal, if an authorized notify operator is logged on, or at the MVS system console if the notify operators are not logged on. These messages are forwarded to the focal point before being displayed.

Use the EZLEASLN routine when you change the resource status, and notification of that change is to be displayed to the operator. The current status in the AON status file is changed by this routine.

The field values passed to EZLEASLN are evaluated against the DDFGENERIC control file entry definitions that guide DDF updates. The following fields, sent to EZLEASLN, are evaluated for DDF generic values:

<table>
<thead>
<tr>
<th>EZLEASLN parameter</th>
<th>DDF generic field</th>
</tr>
</thead>
<tbody>
<tr>
<td>function</td>
<td>func</td>
</tr>
<tr>
<td>status</td>
<td>status</td>
</tr>
</tbody>
</table>
The function value sent is used for the function value for evaluating DDFGENERICT statements.

The parameters passed to the EZLEASLN routine are dependent on the format of the message. Some messages will have more parameters than others. AON messages are shipped in object code format for improved performance. To determine the number of inserts in a message, view the online help for the message. The message inserts are passed as user parms on the call to EZLEASLN.

**Example**

This example shows how to issue a simple status update message. The INACTIVE message has been received for LINE01. The message to be issued, EZL531I, indicates the new status value.

The message text for EZL531I follows:

EZL531I &4 &3 IS INACTIVE DUE TO OPERATOR &5 INTERVENTION

The module to call EZLEASLN is:

```plaintext
USERSLN module
   &CGLOBAL ABC XYZ ... ...
   &TGLOBAL
   *
   * Perform check automation
   *
   EZLECAUT LINE01
   :
   &IF &RETCODE EQ 0 &THEN &GOTO -AUTOOK
   &EXIT
   -AUTOOK
   EZLEASLN NOTIFY=Y,STATUS=Y,AON,INACTV,USERSLN,EZL531,&DATE,
   &TIME,LINE01,LINE,OPER1
   :
   &EXIT
```

The user-written module calls EZLECAUT to determine if automation is valid. If the return code greater than zero (0), automation is not permitted. If it is 0, the routine continues.

After it is determined that NetView automation is in effect, EZLEASLN is called. The function parameter is AON, indicating automation activity triggered the message. Status is INACTV, denoting the new status for LINE01. This status value is placed in the AON status file. The modulename is USERSLN.

The message number EZL531 is suffixed by an I to build a message identifier of EZL531I. EZL531I is inserted at beginning of the message text.

The date and time are coded as &DATE and &TIME.

The resource is coded as LINE01 (VTAM resource name). The resource type is coded as LINE (VTAM resource type), which is &4.
The first user parameter is OPER1. It replaces &5 in the message text.
Checking Thresholds (EZLEATHR)

Purpose

The EZLEATHR routine checks the number of errors recorded in the status file against a preset error threshold. It also supports the recording of the error date and time on the status file.

The EZLEATHR routine searches the automation control file for the applicable threshold of a specific resource. The routine then obtains the error status information from the status file and determines whether any of the three definable thresholds have been exceeded. If a threshold is exceeded, AON issues an error message and an appropriate return code is generated.

This routine can be issued only by another module or by a command processor.

Keep the parameters for the EZLEATHR routine in the same order as they are shown in the following syntax diagram. Use commas or spaces as delimiters.

Syntax

```
EZLEATHR
  resource res_type
  serv_pt
  NEW|CHECK
  OPTION=opt
```

Parameters

`resource`

Specifies the name of the resource for which thresholds should be checked. This resource name can be a generic resource type or any user-defined name up to 8 characters long. The resource name is required and must be the first parameter.

`res_type`

Specifies the type of the resource. The `res_type` parameter can be any user-defined name up to 10 characters long. The resource type is required and must be the second parameter.

`serv_pt`

Specifies the name of the TCP/IP service point controlling the specified resource. If specified, `serv_pt` should be the third parameter. If not specified, the default is the NetView DomainID.

`NEW|CHECK`

Specifies whether a new error should be added to the error status information. If you specify NEW or use the default, an error is added to the error status information, then the thresholds are checked. If you specify CHECK, the thresholds are checked based on the existing error information and no new information is added. If you specify NEW or CHECK, place the parameter in the fourth position.

`opt`

Specifies the AON automation component issuing this request. The list of options follow:
Return Codes

0  No threshold has been exceeded.
1  Infrequent threshold has been reached.
2  Frequent threshold has been reached.
3  Critical threshold has been reached.
4  Incorrect parameters were used in the call.
5  Time-out or other error occurred.

Usage

The EZLEATHR routine accesses the control file to check the THRESHOLD entries and the status file to check the current status of the resource.

You can modify values for the EZLEATHR routine by defining them in the EXIT06 parameter of the ENVIRON EXIT control file entry. Refer to the Tivoli NetView for z/OS Administration Reference for more information. AON uses the values on the ENVIRON EXIT control file entry unless you define a THRESHOLDS control file entry for the specific resource.

The EZLEATHR routine is used primarily to track error conditions that can be recursive. Because EZLEATHR tracks the errors, AON can notify operators of the recursive situation before it causes problems. After a critical threshold exception, AON deactivates resources to stop the recursive error and recovery.

A maximum of 10 errors can be stored in each record in the status file. These errors are stored in order of date and time-of-error. No details about the error are stored.

The VSAM key to the status file is built from the 8-character VTAM resource name. Only one status file record exists for each resource.

Each resource name in the status file must be unique; therefore, select your resource names carefully.

The EZLEATHR routine searches for the threshold in a predefined sequence to find the appropriate threshold. For a description of the threshold search sequence, refer to the Tivoli NetView for z/OS Administration Reference.

AON is designed to fulfill most needs of thresholds automation processing. However, your installation may require you to tailor threshold automation processing. You can code AON user exits for threshold values for a specific resource. These values are specified on the THRESHOLDS statement or on the ENVIRON EXIT control file entry.
Example

This example shows the relationship between the EZLEATHR routine and the automation control file. The threshold to be checked is a user-defined name, PU001. The user module checks the thresholds by calling the EZLEATHR routine.

The automation control file entry is:
```
THRESHOLD PU001,CRIT=(8,02:00),FREQ=(4,04:00),INFR=(4,08:00)
```

The module to call the EZLEATHR routine is:
```
USERTHR module
   &CGLOBAL ABC XYZ ... ...
   &TGLOBAL ABC XYZ ... ...
   EZLECAUT ... check whether automation allowed and set
   ...

-AUTOOK
   EZLEATHR PU001 USER NEW
   &IF &RETCODE EQ 0 &THEN &GOTO -OKSOFAR
   &IF &RETCODE EQ 1 &THEN &GOTO -EXCEEDINFR
   &IF &RETCODE EQ 2 &THEN &GOTO -EXCEEDFREQ
   &IF &RETCODE EQ 3 &THEN &GOTO -EXCEEDCRIT
   otherwise, an error occurred, RC=4/5, log error msg
   &EXIT

-OKSOFAR
   perform whatever actions may be required if no thresholds
   have been exceeded
   &EXIT

-EXCEEDINFR
   perform whatever actions may be required if infrequent
   thresholds have been exceeded
   &EXIT

-EXCEEDFREQ
   perform whatever actions may be required if frequent
   thresholds have been exceeded
   &EXIT

-EXCEEDCRIT
   perform whatever actions may be required if critical
   thresholds have been exceeded.
   &EXIT
```

The THRESHOLD entries in the automation control file defined a critical threshold as 8 errors occurring in 2 hours, frequent threshold as 4 errors in 4 hours and infrequent threshold as 4 errors in 8 hours.

The user-written module initiates the EZLEATHR routine using resource name, resource type, and NEW. The error information is added to the status file and the thresholds are checked.

Upon return to the user-written module the &RETCODE value is checked. If the value indicates that a critical threshold has been exceeded, the module stops recovery, which is consistent with the message issued by the EZLEATHR routine.

Example

This example shows a REXX command list that uses the EZLEATHR routine:
```
/* REXX example of EZLEATHR usage */
/* Check thresholds */
'EZLEATHR 'Resource' USER NEW'
```
Thr_rc = Rc

select
  when Thr_rc = 0 then /* None exceeded*/
    do /* Actions for no threshold exceeded */
      exit
    end
  when Thr_rc = 1 then /* Infrequent */
    do /* Infrequent Actions */
      exit
    end
  when Thr_rc = 2 then /* Frequent */
    do /* Frequent Actions */
      exit
    end
  when Thr_rc = 3 then /* Critical */
    do /* Critical Exceeded, */
      /* terminate recovery */
      exit
    end
  Otherwise
    do
      'MSG LOG, ERROR: RETURN CODE 'Thr_RC' FROM EZLEATHR 'Resource
      exit
    end
end
INFORM Action (EZLECALL)

Purpose

EZLECALL is a REXX routine that generates an immediate INFORM action based on the notification policy. Enter the name of the individual or group policy to contact, and any message text required. EZLECALL does require that an INFORM/CONTACT policy entry exists for this purpose.

The INFORM command is a command synonym (CMDSYN) for EZLECALL.

Syntax

```
EZLECALL policy_name message
```

Parameters

- `policy_name`
  Specifies the name of the INFORM policy or group name to use, which determines the individuals to contact.

- `message`
  Specifies the message to be sent to the contact. The message must be consistent with the CONNECTION type specified in the INFORM policy. If no message is provided, the default INFORM message or the message specified in the INFORM policy is used.

Example

This example shows how the operator can issue an inform for the policy NITEOPS. The text following the policy is sent in an e-mail or to an alpha-pager.

```
INFORM NITEOPS PLEASE CALL THE OFFICE IMMEDIATELY
```

Refer to the [Tivoli NetView for z/OS Administration Reference](#) for additional information on how to setup your INFORM policy statements.
Using Active Monitoring and Recovery (EZLECATV)

Purpose

The EZLECATV routine tries to recover VTAM resources through the VTAM VARY INACT and VARY ACT commands at intervals specified in the MONIT control file entry. Recovery attempts continue until the resource becomes active, interval settings have been exhausted, automation is turned off, or an operator issues a command for that resource (VARY INACT or VARY ACT). If the notify flag for the interval is Y, the EZLECATV routine issues a reminder notification.

EZL507I resname HAS BEEN UNRECOVERABLE FOR duration

At the first interval, message EZL506I issues:

EZL506I restype resname ON location
INACTIVE - RECOVERY MONITORING HAS BEEN INITIATED

Keep the parameters for the EZLECATV routine in the same order as they are shown in the following syntax diagram.

Syntax

EZLECATV

Parameters

resource
Resource name.

res_type
Resource type.

count
MONIT interval count of interval just run. Valid values are 0 through 99.

gmtdate
Date of original failure that initiated recovery. The date is given in Greenwich mean time (GMT).

gmtime
Time of original failure that initiated recovery. The time is given in Greenwich mean time (GMT).

Usage

Link stations and CDRM are not inactivated during recovery. All other resources are deactivated with a VARY INACT,F command to clear pending states. Activations are done with SCOPE=U for PUs and higher nodes. SCOPE=ONLY is used for LUs and CDRMs. Recovery is stopped if the resource becomes unknown to VTAM in consideration of dynamic CDRSCs. A timer is scheduled at the next MONIT interval with an ID of the resource names.

To tailor reminder interval automation processing code AON user exits for specific MONIT intervals for a resource. Set default exit values for EZLECATV on the MONIT statement or on the ENVIRON EXIT control file entry. Refer to the Tivoli
for more information about the ENVIRON EXIT entry.
Checking Automation (EZLECAUT)

Purpose

The EZLECAUT routine checks recovery automation flags on the RECOVERY control file entry to determine whether automation is in effect for a resource.

Syntax

```
EZLECAUT (res_name res_type)
```

Parameters

- `res_name`
  - The name of the resource for which to check automation settings
- `res_type`
  - Type of resource being checked

Return Codes

- 0: Automation is in effect.
- 1: Automation is not in effect.
- 2: Automation is defined, but not in effect due to a NOAUTO window.
- 4: Parameters that are not valid.

Usage

If a NOAUTO parameter is found, calculations determine whether the current time is within the NOAUTO window. The return code is set accordingly.

You can set defaults for the EZLECAUT routine by defining the EXIT07 parameter of the ENVIRON EXIT control file entry. To tailor recovery automation processing, code AON user exits for specific recovery processing.
EZLEFAIL

Processing Generic Failures (EZLEFAIL)

Purpose
The EZLEFAIL routine processes resource failure events and drives option specific routines from the option tables. The following diagram illustrates the syntax of the EZLEFAIL routine and its parameters.

Syntax
```plaintext
EZLEFAIL
       )zb261111)zb261111
       EZLEFAIL
       )zb261111
       RESNAME=resource_name
       RESTYPE=resource_type
       OPTION=option_type
       )zb261111
       TBLKEY=key
       SKIP=
       AUTOSTAT=status
       keyword=value
```

Parameters
- **RESNAME**
  Specifies the resource name.
- **RESTYPE**
  Specifies the resource type.
- **OPTION**
  Specifies the automation component:
  - APPN
  - IP390
  - NVAIX
  - SA
  - SNA
  - SNBU
  - TCPIP
  - X25
- **TBLKEY**
  Specifies unique handling for the event. The EZLEFAIL routine gets optional processing values from the TBLKEY parameter. If you do not specify the TBLKEY parameter for the EZLEFAIL routine, no optional processing or notification occurs. The values on the TBLKEY parameter specify keywords found in the option definition tables. In the option definition table, the keywords define the actual processing values used for optional processing. AON saves the TBLKEY values in the `outmsgid` and `spec_function` variables. Message EZL509I is the default `outmsgid`. The value of TBLKEY is in the following format:
tblkey_value=(outmsgid,spec_function_call)

For example, the EZLEFAIL routine is called with:
EZLEFAIL OPTION=SA MSGPRMS=(OPID) TBLKEY=IST105I RESNAME=resname

The EZLEFAIL routine gets the values specified on the IST105I keyword in the option definition table. In the option definition table, the values on the IST105I keyword are:
IST105I=(EZL531,FKVEAIDA(resname restype))

The EZLEFAIL routine issues the EZL531I message and runs FKVEAIDA as a function sending the current value of resname (resource name) and restype (resource type) for optional processing. An optional processing program would perform automation or processing unique to the resource or failure. No optional processing is done and no message is issued if SKIP=(0) is specified on the EZLEFAIL call.

The EZLEFAIL routine issues message EZL509I to all logs and to DDF. Operators do not receive this message. This message is not issued if SKIP=(A) is specified on the EZLEFAIL call.

**SKIP**
Specifies which processing to ignore:
- **A** Availability message is skipped.
- **C** Check automation processing is bypassed.
- **T** Threshold processing is skipped.
- **O** Option-specific processing (specified by TBLKEY value) is bypassed.
- **M** Option-specific message processing (TBLKEY value) is bypassed.
- **I** Do not gather resource-specific information.
- **R** Recovery attempt processing is bypassed. Setting the AIP operator status is also bypassed.
- **H** Do not check higher node status.

**AUTOSTAT**
Specifies the status.

*keyword=value*
Any valid keyword and value.

**Return Codes**
- **00** EZLEFAIL completed successfully.
- **02** AON initialization has not completed, issue EZL003E.
- **03** Missing parameters, issue EZL203I.
- **04** Incorrect parameters: issue EZL204I.
- **05** Wait time expired: issue EZL205I.
- **06** Command failed: issue EZL206I.
- **07** NOVALUE variable found.
- **08** REXX syntax failure.
- **09** Initialization shipped to automation operator.
- **10** Could not find resource type/owning option.
- **11** Check recovery, option not enabled.
- **12** Check recovery, higher node down.
- **13** Check recovery, automation flag off.
EZLEFAIL

14 Check recovery, resource in available state.
15 Check recovery, recovery timer exists.
16 Automation is defined, but not in effect due to a NOAUTO window.
21 Check thresholding: infrequent routine RC>0.
22 Check thresholding: frequent routine RC>0.
23 Check thresholding: critical routine RC>0.
30 Optional processing routine RC>0.

Example

The following example illustrates how you can call EZLEFAIL from the automation table:

```
IF MSGID='IST285I'
  & TEXT = . 'DUMP OF ' RESNAME ' FAILED - PERMANENT' .
  & HDRM'TYPE = 'Q'
THEN
  EXEC(CMD('EZLEFAIL OPTION=SA RESTYPE=NCP'
    ' RESNAME=' RESNAME
    ' SKIP=(C,T,R)'
    ' TBLKEY=IST285I AUTOSTAT=NCPDUMP')
  ROUTE(ALL *));
```

The automation flag is not checked, threshold analysis is not performed, and recovery is not initiated. The status sent to DDF is NCPDUMP and IST205I. A keyword in the option definition table specifies any additional message or program to be called.

Example

This example shows AON/SNA automation being driven for VTAM message IST105I when a resource fails. EZLEFAIL is called out of the automation table for the AON/SNA subarea (SA) option. Other parameters passed to EZLEFAIL include TBLKEY that identifies an option definition table key to use for recovery of the resource. Additional Information is retrieved from the option definition table during the failure processing.

```
IF TEXT = 'IST105I' resname 'NODE' . & HDRM'TYPE = 'Q'
THEN
  EXEC(CMD('EZLEFAIL OPTION=SA MSGPRMS=(OPID) '
    'TBLKEY=IST105I '
    'RESNAME=' resname));
```

Example

This example shows AON/SNA automation is driven for VTAM message IST619I when a resource fails. EZLEFAIL is called out of the automation for the AON/SNA subarea option. The option definition table key for additional recovery information and processes is IST619I. Additional information is retrieved from the option definition table during the failure processing.

```
IF MSGID='IST619I'
  & TEXT = . 'ID='resname ' FAILED' .
  & HDRM'TYPE = 'Q'
THEN
  EXEC(CMD('EZLEFAIL OPTION=SA TBLKEY=IST619I '
    ' RESNAME=' resname )
    'SKIP=(R) ROUTE(ALL *));
```
Managing Automation Tables (AUTOCMD/EZLEF002)

Purpose

The AUTOCMD routine is used to enable, disable, insert, and get the status of automation tables. The following diagram illustrates the syntax of the AUTOCMD routine and its parameters.

Syntax

AUTOCMD

Statement:

ENABLES, tablename, labeltype, labelname

DISABLES

Single Table:

ENABLE, tablename

DISABLE

REMOVE

STATUS

Insert:

INSERT, tablename, position, marker, listname

SWAP:

SWAP, tablename1, tablename2, marker, listname
SWAPI:

SWAPI-, tablename-, tablename2, marker?, listname

Status:

STATUS-, tablename-

Statusm:

STATUSM-, marker-

Command | Synonym
---------|----------
AUTOCMD  | EZLEF002

Parameters

ENABLES
Activates multiple automation tables.

DISABLES
Deactivates multiple automation tables.

ENABLE
Activates a single automation table.

DISABLE
Deactivates a single automation table.

INSERT
Loads an automation table in the position specified.

SWAP
Replaces an automation table with another automation table.

SWAPI
Replaces an automation table with another automation table and reestablishes all disabled elements.

STATUS
Returns the status of the specified table.

STATUSM
Returns the status of the table with the specified marker.

tablename
Specifies the name of the automation table to be affected by the chosen keyword.

tablename2
Specifies the name of the automation table that will replace tablename.
**labeltype**

Specifies the type of label to be altered as follows:

- LABEL
- ENDLABEL
- BLOCK
- GROUP
- SEQUENCE

**labelname**

Specifies a label name, which consists of 1–16 character label or 1–8 character sequence numbers in the automation table.

**position**

Specifies the position of the table to be loaded. The position specification follows:

- **1..n** The numerical position of the table.
- **FIRST** The table loads in the first position.
- **LAST** The table loads in the last position.

**marker**

Specifies a unique identifier associated with the table being loaded. Markers can be 1–8 characters in length and are optional.

**listname**

Specifies a unique name for the listing member associated with the automation table. A list name can be 1–8 characters. If the listing name is omitted, a unique name will be created and a listing is generated, automatically. However, if *NONE* is provided as a parameter, then no listing is generated. This is considered an override and is not recommended. No informational messages are generated regarding an override of the listing with *NONE*.

**Return Codes**

For STATUS and STATUSM, the return codes are as follows:

- **0** The search for the automation table or marker was successful.
- **4** The search for the automation table was not successful or an error occurred.

For all other return codes are as follows:

- **0** The request completed successfully.
- **n** The request failed. The error message can be found in the common global variable EZLERRMSG. The common variable stem EZLAUTOR contains messages generated by the request.

**Usage**

When loading or swapping tables that replace or become the automation table used by AON, it is necessary to update the EZLMSGTBL and EZLMSGLST global variables with the table name and listing name respectively. If AUTOMAN removes the AON table, then these variables are automatically reset.
Setting Panel Message Color (EZLEMCOL)

**Purpose**
Use EZLEMCOL to set panel variable attributes for non-AON messages such as VTAM and NetView messages.

**Syntax**

```
EZLEMCOL (msgnum, msgtext)
```

**Parameters**
- `msgnum` Message number
- `msgtext` Message text

**Usage**
EZLEMCOL can also be called as a command.

Color settings are:
- **Information message**: White
- **Warning message**: White
- **Error message**: Yellow
- **Action message**: Red

**Example**
Following is a sample of REXX code using EZLEMCOL:

```
$ErrorMsg = EZLEMCOL(MsgId, MsgTxt) /* set panel variable */
'GLOBALV GETT EZLMSGCOLR' /* get color attribute(s) */
$ ERRORMSG = 'EZLMSGCOLR' /* set panel attribute(s) */
```
EZLEMSG

Formatting Panel Messages (EZLEMSG)

Purpose

EZLEMSG is a REXX function that builds messages to be sent to operators while they are viewing an AON panel.

Syntax

\[
\text{EZLEMSG} \quad (\text{msgnum}, \text{logupdate}, \text{clistname}, \text{parm1}, \text{parm2}, \ldots, \text{parmn})
\]

Parameters

- **PnlMsg**
  The panel variable used to display the message. Upon return from EZLEMSG this variable is set to the complete message text with inserts included.

- **msgnum**
  Identifies the message to display.

- **logupdate**
  If Y, the message is passed back from the function call and is also be logged to the NetView log. If N, the message is returned from the function call and is not logged.

- **clistname**
  Used to identify the program issuing the message. This is required and issued if an error occurs to properly identify the failing program.

- **parm1**
  First insert for the message. This is optional.

- **parm2**
  Second insert for the message. This is optional.

- **parmn**
  Last insert for the message. This is optional.

Usage

The default message prefix is EZL. If you generate messages with other prefixes, code the prefix plus the message number. The parameters (parm1 through parmn) are optional and are only required if the message substitutes variable information into the message text as a message insert.

Example

Following is a sample of REXX code using EZLEMCOL:

\[
PnlMsg = \text{EZLEMSG}(901,'N',\text{ident},\text{sel},'1','5')
\]

This statement sets the variable PnlMsg equal to the text of message EZL901I with inserts of the value of variable sel, 1, and 5:

\[
PnlMsg = \text{SELECTION 6 IS INVALID. TYPE A NUMBER BETWEEN 1 AND 5}
\]
EZLENFRM

Driving the Inform Policy (EZLENFRM)

Syntax

EZLENFRM

- policy_name
- resname
- restype
- resdomain
- resstat

- aostat

Purpose

EZLENFRM is a REXX routine called by the AON notification policy when an inform action is required. EZLENFRM can also be invoked by non-AON routines. EZLENFRM checks the inform policy, issues the inform actions, and logs the action, if indicated. All parameters are required and can be substituted in the resulting inform message.

Parameters

policy_name
- Specifies which INFORM policy or group name to use when determining who to contact.

resname
- The failing resource name

restype
- The failing resource type

resdomain
- The domain of the failing resource

resstat
- The status of the failing resource

aostat
- Automation status of the failing resource

Example

This example shows an invocation of EZLENFRM for policy NITEOPS. A CDRM that is down has reached the end of its monitoring intervals. The NITEOPS policy will be consulted and appropriate actions will be taken.

EZLENFRM NITEOPS NTB6MVS CDRM PACDR NOMOMONS

Refer to the [Tivoli NetView for z/OS Administration Reference](#) for more information about setting up INFORM policy statements.
Notify Policy List (EZLENTFY)

Purpose

EZLENTFY is a REXX routine called to query the AON Notify Policy for a given event type, resource, or resource type.

Syntax

```
EZLENTFY
```

```
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>event_type</td>
<td>Query Notify Policy based on the type of event such as CRITTHRS.</td>
</tr>
<tr>
<td>resname</td>
<td>Query Notify Policy based on the resource name. Wildcards are supported.</td>
</tr>
<tr>
<td>restype</td>
<td>Query Notify Policy based on the resource type.</td>
</tr>
</tbody>
</table>

Parameters

- `event_type`: Query Notify Policy based on the type of event such as CRITTHRS.
- `resname`: Query Notify Policy based on the resource name. Wildcards are supported.
- `restype`: Query Notify Policy based on the resource type.

Usage

If any Notify Policy keywords are set to `NO`, they do not appear in the Notify Policy List. Refer to the [Tivoli NetView for z/OS Administration Reference](#) for additional information on the NOTIFY statement.

The Notify Policy List can be null. Any caller must be prepared to avoid issuing notifications with a null Notify Policy List.

The following INFORM Policy Names are not valid:
- MSG
- ALERT
- TEC
- DDF

EZLENTFY creates two task global variables:
- `EZLNTFYLIST` containing the list of notification actions. The format of the list is `(Notify1, Notify2, ..., NotifyN)`
- `EZLNTFYTYPE` containing the NOTIFY policy statement used.

User Exit (EXIT10) Functions:
- Override NOTIFY policy based on other criteria, such as your own notify policy database.
- Implement other notifications such as fax.
Setting the AIP User Status Bit (EZLERAIP)

Purpose
EZLERAIP is a REXX routine that enables the setting and resetting of the Automation in Progress (AIP) user status bit for resources being monitored or viewed in NMC. This routine handles the routing of the request to the domain specified in the control file. On the target domain, EZLERAIP routes the request to the appropriate AUTAIP autotask for processing. This routine can also update the Operator Intervention View (OIV) status, see the request types (reqtype) for specific information.

Syntax

```
EZLERAIP

EZLERAIP resname, restype, reqtype, other_netid, nqn_netid,

prev_domain, gw_flag, sscp_name
```

Parameters

```
resname
The name of the resource.

restype
The restype parameter must be one of the following:
- CDRM
- LINE
- LINKSTA
- NCP
- PU

reqtype
The operation to be performed. The following values are supported:
SET Sets the AIP status for the resource.
RESET Resets the AIP status for the resource.
SET/OIV Sets the AIP status and removes the resource from the Operator Intervention View.
RESET/OIV Resets the AIP status and adds the resource to the Operator Intervention View.

other_netid
The NETID of the resource if owned by another domain.

nqn_netid
The NETID of a resource as specified in its network qualified name.
```
prev_domain
An internal parameter that is filled in by the EZLERAIP call when it reroutes the request to the specified RODMDOM. All other calls should omit this parameter.

gw_flag
When set to Y causes the AIP operator status to update for both the NCP and Gateway NCP resources. Otherwise, only the NCP resource changes.

sscp_name
The VTAM SNA node name of the reporting VTAM. This is required only when the resource is owned by another VTAM domain.

Usage

• Commas are required between each of the parameters.
• For LNM resources, the first three parameters are required.
• For bridges, the resname must be entered in the following format: LNMname.bridgenname.
Routing Commands over Cross-Domain Sessions (EZLERCMD)

Purpose
EZLERCMD is a REXX routine that routes commands over cross-domain sessions. Both NetView-NetView Task (NNT) and remote command (RMTCMD) session types are supported.

Syntax
EZLERCMD

Parameters
- **domain_id**
  Domain to which the command is sent.
- **operator_id**
  Operator to issue the command. This parameter is for RMTCMD sessions only.
- **NNT**
  Specifies that the session the command is routed over is an NNT session.
- **RMTCMD**
  Specifies that the session the command is routed over is an RMTCMD session.
- **NONCCF**
  If specified, the command is issued following CMD HIGH. Otherwise, it is issued following CMD HIGH NCCF.
- **command**
  The command and parameters to be routed.

Usage
The **operator_id** variable is required for RMTCMD sessions, but not for NNT sessions. No verification is performed on the command. It is routed as is to the target domain. If the specified operator is not logged onto the target domain, NetView establishes an RMTCMD session and issues the command upon session initialization. Ensure that the specified operator is logged on to the target domain before calling EZLERCMD.

Example
This example routes the LIST STATUS=OPS command over the active NNT session to domain CNM01.

EZLERCMD CNM01,,NNT,,LIST STATUS=OPS

Example
This example shows how to route the LIST STATUS=OPS command over the RMTCMD session to OPER1 on domain CNM01.

EZLERCMD CNM01,OPER1, RMTCMD,, LIST STATUS=OPS
Recovering Resources (EZLERECV)

Purpose

The EZLERECV routine is called for resource recovery and drives option specific routines from the option tables. The following diagram illustrates the syntax of the EZLERECV routine and its parameters. The AIP operator status is also cleared each time EZLERECV is called for a given resource.

Syntax

```
EZLERECV
```

The EZLERECV routine gets optional processing values from the TBLKEY parameter. If you do not specify the TBLKEY parameter for the EZLERECV routine, no optional processing or notification occurs. The values on the TBLKEY parameter specify keywords found in the option definition tables. In the option definition table, the values on the IST093I keyword are:

For example, if the EZLERECV routine is called with:

```
EZLERECV OPTION=SA MSGPRMS=(OPID) TBLKEY=IST093I RESNAME=resname
```

The EZLERECV routine gets the values specified on the IST093I keyword in the option definition table. In the option definition table, the values on the IST093I keyword are:
**EZLERECV**

The EZLERECV routine issues the EZL517I message and runs FKVEAIDA as a function sending the current value of resname (resource name) and restype (resource type) for optional processing. An optional processing program performs automation or processing unique to the resource or failure. No optional processing is done and no message is issued if SKIP=(0) is specified on the EZLERECV call.

The EZLERECV routine issues message EZL504I to all logs and to DDF. Operators do not receive this message. This message is not issued if SKIP=(A) is specified on the EZLERECV call.

**SKIP**

Specifies which processing to ignore:

- **I**: Do not gather resource-specific information.
- **A**: Skip availability message.
- **R**: Bypass recovery stop processing.
- **S**: Skip active monitoring restart.
- **O**: Bypass option-specific processing (specified by TBLKEY value).
- **M**: Bypass option-specific message processing (TBLKEY value).

**keyword=value**

Any valid keyword and value.

**Return Codes**

- **00**: EZLERECV completed successfully.
- **03**: Missing parameters, issue EZL203I.
- **04**: Incorrect parameters, issue EZL204I.
- **05**: Wait time expired, issue EZL205I.
- **06**: Command failed, issue EZL206I.
- **07**: NOVALUE variable found.
- **08**: REXX syntax failure.
- **09**: Initialization shipped to automation operator.
- **10**: Could not find resource type.
- **11**: Option not enabled.
- **30**: Optional processing routine RC>0.

**Example**

This example shows that AON/SNA automation is driven for VTAM message IST093I when a resource is reactivated. EZLERECV is called from the automation table for the AON/SNA SubArea option with optional parameters, such as TBLKEY=IST093I, to identify additional information for the resource recovery actions from the AON/SNA option definition table. Additional information is retrieved from the option definition table during the recovery processing.

```pascal
IF MSGID='IST093I'
  & TEXT = 'IST093I ' resname 'ACTIVE'
THEN
  EXEC(CMD('EZLERECV OPTION=SA TBLKEY=IST093I ' 'RESNAME=' resname ' MSGPRMS=(OPID)')
    ROUTE(ALL *));
```

178 Automated Operations Network Customization Guide
Routing Commands to Other NetView Domains (EZLERGWY)

Purpose

EZLERGWY is a REXX routine that uses the RMTCMD command to route commands to other NetView domains.

Syntax

EZLERGWY

```
EZLERGWY targ_dom, GATOPER autotask, targ_op, safe_name, request
```

Note: If `safe_name` is coded and contains data, `request` is optional.

Parameters

- **targ_dom**
  The target NetView domain to which to route the request.

- **targ_op**
  The target NetView operator ID to which to route the request. If not specified, the request is routed to the GATOPER autotask as defined in the CDLOG statements in EZLCFG01 in DSIPARM.

- **safe_name**
  The name of the safe in which to store the response or from which to get the command. If you specify DSILOG, the output is only logged to DSILOG. If not specified, the command is routed and no response is logged or returned to the caller.

- **request**
  The command to run.

Usage

To enable the remote gateway support, define CDLOG statements for the GATOPER autotask in each NetView domain. These statements are located in member EZLCFG01 in DSIPARM. Using these CDLOG definitions, AON establishes a RMTCMD session between the corresponding NetView domains. For more information, refer to the *Tivoli NetView for z/OS Administration Reference*.

All parameters are positional and must be delimited by a comma. If you omit a parameter, use a comma to denote its absence.

No syntax checking or security checking is performed for the requested command.

To see the output from your command, specify a `safe_name` parameter. For debugging, specify DSILOG for the `safe_name` parameter.

If you are passing a command in the named safe (`safe_name`) parameter, make sure your program has placed data (the command) into the safe. If not, you will receive message EZL203I. In this case, you should not pass the command as a parameter.
Example

The following causes the LIST command to be routed to domain NTV6D with the output being placed into a named safe called MYSAFE:

EZLERGWY NTV6D,,,LIST

Since no \textit{targ\_op} was specified, GATOPER autotask in NTV6D will be used.

Example

The following causes the LIST command to be routed to domain NTV6D. No output is returned to the calling program:

EZLERGWY NTV6D,,,LIST

Example

The following shows how to invoke EZLERGWY when the command to run is contained in a safe. In this case, the command is \texttt{LIST DSILOG} and is passed to EZLERGWY in a safe called MYSAFE. EZLERGWY builds the command, runs it, and returns the responses back into MYSAFE:

\begin{verbatim}
... Command.0=2 Command.1='LIST ' Command.2='DSILOG' "PIPE STEM Command.|Safe MYSAFE"
"EZLERGWY NTV6D,,MYSAFE,"
"PIPE SAFE Mysafe|Stem Myvar." Do i=1 to myvar.0 Say 'Response to command=' Myvar.i End
\end{verbatim}

Using this example, you can send commands longer than 255 bytes (NetView command line restriction) to remote NetView domains.

Example

The following shows how to start a remote session from the GatOper (GATNTV6D) in domain NTV6D to domain NTVFE as RMTNTV6D:

\begin{verbatim}
CDLOG GATNTV6D,NTVFE,
  SESSTYPE=RMT,
  TARGOP=RMTNTV6D,
  INIT=YES,
  DESC='RMTCMD GATEWAY TO NTVFE'
\end{verbatim}

TARGOP is optional for CDLOG and defaults to the same task name in the target domain. GATNTV6D should already be known to domain NTVFE as the NNT gateway task. INIT=YES should be specified to start the session every time GATNTV6D logs on. DESC= is optional.
Routing NNT Cross-Domain Logon Information (EZLEROUT)

Purpose
EZLEROUT is a REXX routine that routes cross-domain logon information for NNT sessions and automates replies to NetView message DSI809A. You can also use EZLEROUT to route commands over RMTCMD sessions, similar to the EZLERCMD routine.

Syntax

```
EZLEROUT

  domain_id, target_oper, password

  NNT, profile, hcl_device,
  RMTCMD,, command
```

Parameters

- **domain_id**: The domain to which you are logging on.
- **target_oper**: The operator ID to which you are logging on.
- **password**: The password to use.
- **NNT**: Specifies that you are logging onto an NNT session.
- **profile**: The NNT session profile.
- **hcl_device**: The hard copy log device for the NNT session.
- **RMTCMD**: Specifies that you are logging onto an RMTCMD session.
- **command**: The command to be issued. Specifying YES for an NNT session runs the initial program defined in the operator logon profile, NNT-Profile. For RMTCMD sessions, this can be any valid NetView command or program (command list).

Example
This example shows how to route the LIST STATUS=OPS command over the active NNT session to domain CNM01. The routine uses the password OPERWORD, and the operator profile OPERPROF. The double-commas (,,) preceding the LIST STATUS=YES command indicate that the routine should ignore the hard copy log device parameter.

```
EZLEROUT CNM01,,NNT,OPERWORD,OPERPROF,,LIST STATUS=OPS
```
Example

This example shows how to route the LIST STATUS=OPS command over the RMTCMD session to OPER1 on domain CNM01.

EZLEROUT CNM01,OPER1,,RMTCMD,,,LIST STATUS=OPS
Retrieving AON Information (EZLERTVE)

Purpose
The EZLERTVE routine retrieves data from AON option definition tables to use in automation modules.

Syntax
The following diagram illustrates the syntax of the EZLERTVE routine:

EZLERTVE

Parameters
EZLOPT
Specifies the option (AON feature) common global variables from the option definition tables.

EzlRT
Specifies the resource type common global variables from the option definition tables.

option
Installed AON feature or suboption. Valid option names are:

- AON  AON base
- APPN  Advanced Peer-to-Peer Networking (APPN) suboption for AON/SNA
- IP390  TCP/IP for MVS suboption of AON/TCP
- NVAIX  NetView for UNIX suboption for AON/TCP
- SA  Subarea suboption for AON/SNA
- SNA  AON/SNA automation feature
- SNBU  Switched network backup (SNBU) suboption for AON/SNA
- TCPIP  AON/TCP automation feature
- X25  X.25 suboption for AON/SNA

resource_type
Any valid resource type supported by AON or its automation features.

entry
Option definition table keyword.

Usage
The EZLERTVE routine must be called from a program, for example, REXX. The results are returned back to the calling program in a local variable, EZLERTVE.

The EZLERTVE routine accesses a variable name of EZLOPT.option.entry or EZLRT.resource_type.entry and returns the value of the common global variable.
If no value is found, EZLERTVE returns \textit{N/A}.

**Example**

The following is an example of EZLERTVE in a REXX procedure:

```rexx
opt_id=SNA
Restype=PU

/* Get message class for option and resource type */
EZLERTVE EZLOPT opt_id 'MSGCLASS'
opt_class = EZLERTVE
EZLERTVE EZLRT Restype 'MSGCLASS'
rt_class = EZLERTVE

if rt_class = "N/A" then
  rt_class = "00"
if opt_class = "N/A" then
  opt_class = "00"
```

The EZLERTVE routine gets the variable EZLOPT.SNA.MSGCLASS and the variable EZLRT.PU.MSGCLASS.
Issuing Resource State Reminders (EZLESRMD)

Purpose
The EZLESRMD routine checks the status of the resource and issues notifications if the resource is down. The resource is not recovered.

Syntax
```
EZLESRMD

resource resource, res_type res_type, count count, gmtdate gmtdate, gmttime gmttime
```

Parameters
- **resource**
  Resource name.
- **res_type**
  Resource type.
- **count**
  MONIT interval count of the interval just run. Valid values are 0 through 99.
- **gmtdate**
  Date of the original failure that initiated recovery. The date given is in Greenwich mean time (GMT).
- **gmttime**
  Time of the original failure that initiated recovery. The time given is in Greenwich mean time (GMT).

Usage
A timer is scheduled at the next MONIT interval that has the ID of the resource name.

Monitoring is discontinued if the resource becomes unknown to VTAM in consideration of dynamic CDRSCs.

Monitoring continues until the resource becomes active, interval settings have been exhausted, or automation is turned off. If the notify flag for the interval is Y, a notification is issued:

```
REMITTER - restype resname HAS BEEN DOWN FOR interval
```

At the first interval, message EZL506I is issued:

```
resname INACTIVE - ATTEMPTING RECOVERY - RECOVERY MAY BE TERMINATED BY VARYING THE RESOURCE INACTIVE
```

The EZL540I message is issued when reminders are halted (REMINDERS FOR restype resname HALTED - REMINDER THRESHOLDS UNDEFINED).
EZLESTOP

Stopping Cross-domain Sessions (EZLESTOP)

Purpose
EZLESTOP is a REXX routine supplied with AON to stop cross-domain NetView sessions. Both NNT and RMTCMD sessions are supported.

Syntax

```
EZLESTOP
```n
```
EZLESTOP domain_id,
```n
```
target_oper
```

Parameters

- **domain_id**
  Domain with which you have a session.
- **target_oper**
  Operator to be logged off. This is valid only for RMTCMD sessions.

Usage
EZLESTOP determines the type of session based on the parameters that are provided. If the target operator is specified, then EZLESTOP looks for a RMTCMD session and logs that operator off. If you do not specify a target operator, EZLESTOP stops your NNT session to the specified domain.

Return Codes

- **0**  
  Session was ended.
- **4**  
  Unable to stop cross-domain session. Check task global variable, EXITMSG, for the message that was received.
- **5**  
  Internal AON error. Call Tivoli customer support.
- **7**  
  Internal AON error. Call Tivoli customer support.
- **8**  
  Internal AON error. Call Tivoli customer support.

Example
This example logs operator OPER2T off a remote command (RMTCMD) session on domain CNM02.

```
EZLESTOP CNM02,OPER2T
```

Example
This example shows how to log off your NetView-NetView task (NNT) session with domain CNM02.

```
EZLESTOP CNM02
```
Starting Cross-domain Sessions (EZLESTRT)

Purpose

EZLESTRT is a REXX routine that starts cross-domain NetView sessions. The routine supports both NNT and RMTCMD type sessions.

Syntax

```
EZLESTRT

EZLESTRT—domain_id,—target_oper,—password,

EZLESTRT—NNT,profile,hard_copy_device,logmode,—initial_cmd

EZLESTRT—RMTCMD,,,
```

Parameters

- **domain_id**: Target domain to start session.
- **target_oper**: Operator to be logged on.
- **password**: NNT password.
- **NNT**: Start a NetView-to-NetView Task session.
  - **profile**: NNT operator profile.
  - **hard_copy_device**: NNT hard—copy log device to start.
  - **logmode**: Log mode for NNT session.
- **RMTCMD**: Start a remote command session.
  - **initial_cmd**: Initial program or command to issue. For an NNT session, YES runs the initial program that is defined in the operator’s logon profile. For RMTCMD sessions, this can be any valid NetView command or program.

Usage

Delimit all parameters with commas. If you do not specify a parameter, use two commas.

Return Codes

- **0**: Cross-domain session started.
- **4**: Unable to start cross-domain session. Check the task global variable, EXITMSG, for the message that was received.
- **5**: Internal AON error. Call Tivoli customer support.
- **7**: Internal AON error. Call Tivoli customer support.
Internal AON error. Call Tivoli customer support.

Example

This example shows how to start a cross-domain RMTCMD session to CNM02 as OPER2T:

```
EZLESTRT CNM02,OPER2T,,RMTCMD,,,,OPER2IC
```

The first occurrence of double-commas (,,) means that no password will be passed. Other parameters not being passed are the profile, hard_copy_device, and logmode.

Example

This example shows how to start a cross-domain NNT session to CNM02 as OPER1 using a password of OPERPSWD and a logon profile of OPERPROF:

```
EZLESTRT CNM02,OPER1,OPERPSWD,NNT,OPERPROF,,,YES
```

No parameters are supplied for hard_copy_device and logmode. YES specifies to run the operator’s initial program as defined in the profile OPERPROF when the session is established.
Activating VTAM Resources (EZLEVACT)

Purpose

Use the EZLEVACT REXX routine to reactivate VTAM resources. EZLEVACT issues a VARY NET,ACT command for a specified resource.

Syntax

```
EZLEVACT
```

```r
EZLEVACT resource
```

Parameters

`resource`

The name of the VTAM resource to be activated.

Return Codes

- **0**: VARY ACT was successful.
- **8**: WAIT error or resource is not known.
- **9**: Security failure.
- **100**: WAIT timeout.

Usage

EZLEVACT can be called from any NetView program or command processor.

Example

The following example shows how to activate resource TA1P523A:

```
EZLEVACT TA1P523A
```
Deactivating VTAM Resources (EZLEVINA)

Purpose
Use the EZLEVINA REXX routine to deactivate VTAM resources. EZLEVINA issues a VARY NET,INACT,F command for a specified resource.

Syntax
EZLEVINA

Parameters
resource
The name of the resource to deactivate.

Return Codes
0  VARY ACT was successful.
8  WAIT error or resource is not known.
9  Security failure.
100  WAIT timeout.

Usage
EZLEVINA can be called from any NetView program or command processor.

Example
The following example shows how to deactivate resource TA1P523A:
EZLEVINA TA1P523A
Moving VTAM Resources (EZLEVMOV)

Purpose
Use the EZLEVMOV REXX routine to move a PU from one line to another. EZLEVMOV issues a MODIFY NET,DR,TYPE=MOVE,FROM=xxxx,TO=yyyy command for a specified resource.

Syntax
```
EZLEVMOV

EZLEVMOV PU_name,from_line,to_line
```

Parameters
- **PU_name**
  The name of the PU to move
- **from_line**
  The line to which the PU is currently attached
- **to_line**
  The line to which to move the PU

Return Codes
- **0** MOVE was successful.
- **8** WAIT error, MOVE error, incorrect number of parameters.
- **9** Security failure.
- **100** WAIT timeout.

Usage
EZLEVMOV can be called from any NetView program or command processor. If you choose to move a PU, reactivate it after the move is complete. You can use the EZLEVACT for the reactivation.

Example
The following example shows how to move PU TA1P523A from line TA1L5023 to line TA1L5024:
```
EZLEVMOV TA1P523A,TA1L5023,TA1L5024
```
EZLSMSU

Sending MSUs to an MS Transport Application (EZLSMSU)

Purpose

EZLSMSU can be used to issue the NetView MS Transport send macro. To use EZLSMSU you must build a CP_MSU major vector and then invoke EZLSMSU to issue the send macro.

Syntax

```
EZLSMSU
```

```
EZLSMSU
   \[\textbf{VARNAME=}\texttt{varname}\]
   \[\textbf{DESTAPPL=}\texttt{destappl}\]
   \[\textbf{DESTLU=}\texttt{destlu}\]
   \[\textbf{ORIGAPPL=}\texttt{origappl}\]
```

Parameters

varname
Identifies the variable that contains a prebuilt CP_MSU. The variable is retrieved from your local variable pool. EZLSMSU does not perform data verification; it assumes that the CP_MSU was built correctly.

destappl
Identifies the destination MS application for the send. If you do not specify a destination, the default (ALERT) is used and the MSU is sent to the NetView hardware monitor.

destlu
Identifies the destination MS LU name for the send. For NetView applications, this is the NetView domain ID. If you do not specify a name, the default of the current NetView domain is used.

origappl
Identifies the application originating the send request. If you do not specify an application, the default (AONALERT) is used.

Usage

No checking is done on your MSU data. All major vectors, subvectors, and subfields must be built properly. For more information refer to the SNA formats. The data sent to the hardware monitor (DESTAPPL=ALERT) can be an alert, a resolution, or an MSU containing multiple alerts or resolutions. To be notified when a send fails, code an MS Transport application and specify it for the ORIGAPPL. AON registers the AONALERT MS application during initialization. To see an example of how to build a CP_MSU acceptable for EZLSMSU, browse the EZLEMSU AON program.
Return Codes

EZLSMSU returns RC=0 if the MSU was sent to the NetView MS Transport. If an error occurs, EZLSMSU returns the return code received from NetView. The return codes are documented in *Tivoli NetView for z/OS Customization: Using PL/I and C*.

Example

The following example shows how to send the data built in the variable MSUDATA to the ALERT application (hardware monitor) running in your domain from the AONALERT application:

```plaintext
EZLSMSU VARNAME=MSUDATA
```

Example

The following example shows how to send the MSU data contained in the variable MSUDATA to the MYAPPL application in domain CNM1A from the MYAPPL application in your current NetView domain:

```plaintext
EZLSMSU VARNAME=MSUDATA,DESTAPPL=MYAPPL,DESTLU=CNM1A,ORIGAPPL=MYAPPL
```
Running Entry and Exit Traces (EZLTRACE)

Purpose

Use the EZLTRACE routine in modules and command processors to:

- Provide entry and exit tracing
- Control more detailed levels of tracing

Also, use this routine in your automation modules. Tracing is controlled at the operator panel.

Syntax

EZLTRACE

ENTRY command parm

EXIT

ENTRY

Indicates start of command

EXIT

Indicates end of command

command

The module being traced

parameters

Input parameters passed to command (entry)

return_data

Return values or return code (exit)

Usage

The parameters variables are written to the log when EZLTRACE ENTRY is run. The return_data variable is written to the log when EZLTRACE EXIT is run.

Local variables are set for use by either REXX or NetView command list language for detail tracing:

- EZTRACEC - NetView command list program
- EZTRACER - REXX
- Entry message
- Exit message

Example

Use the EZLTRACE routine in a REXX procedure to trace entries:

parse source . Invoc Ident .
parse upper arg argstring
'EZLTRACE ENTRY Ident argstring /* Entry trace */
Trace Value(EZLTRACER)

Example

Use the EZLTRACE routine in a REXX procedure to trace exits:

ReturnCode = 4
'EZLTRACE EXIT Ident ReturnCode /* Exit trace */
exit ReturnCode
Example

Use the EZLTRACE routine in AON to avoid calls to EZLTRACE when trace is set to NONE:

```
parse source . Invoc Ident .
parse upper arg argstring
'Globalv Getc EZLTRACED'
If Substr(EZLTRACED,1,4) <> 'NONE'
Then Do 'EZLTRACE ENTRY 'indent argstring
Trace Value(EZLTRACER)
End
```

Example

Use the EZLTRACE routine in a NetView command list to trace entries:

```
MYCODE CLIST
 &CONTROL ERR
 &IDENT = MYCODE
 * EZLTRACE ENTRY &IDENT &PARMSTR
 &CONTROL &EZLTRACEC
 *
```

Example

Use the EZLTRACE routine in a NetView command list to trace exits:

```
* &RETCODE = 99
EZLTRACE EXIT &IDENT &RETCODE
 &EXIT &RETCODE
*```
SNMP RFC Conversion (FKXECNVT)

Syntax
Use the FKKXECNVT routine to read SNMP MIB RFCs and create entries that can be pasted into /etc/mibs.data.

```
FKXECNVT
     indname outddname mibddname
```

Parameters

- **indname**: Specifies the dataset pointer to the input SNMP MIB RFC.
- **outddname**: Specifies the dataset pointer of where to put the output mibs.data entries.
- **mibddname**: Specifies the dataset pointer to the MIB OBJECT file.

Usage
The input file to FKKXECNVT must be in textual format. FKKXECNVT reads the MIB RFC line by line, and will not handle files stored in string format. Maximum LRECL for all files used by FKKXECNVT is 256.

Example

```
FKXECNVT USER1.RFCDATA(TN3270E) USER.RFCDATA(TN32OUT)
NETVIEW.V5R1M0.CNMSAMP(FKXMOBJ)
```

This converts the MIB file named TN3270E to a mibs.data file called TN32OUT and uses FKXMOBJ from the NetView sample library as the base line for the conversion.

Messages
Following are error messages that may occur with the FKKXECNVT function.

<table>
<thead>
<tr>
<th>Error</th>
<th>Explanation</th>
<th>Operator Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>No Files</td>
<td>No parameters were entered for this invocation.</td>
<td>Retry the command specifying the Input, Output, and MIB files.</td>
</tr>
<tr>
<td>No MIB Data File name was entered.</td>
<td>Parameters were not entered for this invocation.</td>
<td>Retry the command specifying the Input, Output, and MIB files.</td>
</tr>
<tr>
<td>No Output File</td>
<td>An output file name was not entered.</td>
<td>Retry the command specifying the Input, Output, and MIB files.</td>
</tr>
<tr>
<td>Unable to allocate Input file indsn Return Code = RC</td>
<td>The ALLOC command failed for file indsn.</td>
<td>RC is the return code from the ALLOC command. Use this to determine why the allocate failed and resolve the problem and then retry the command.</td>
</tr>
</tbody>
</table>
Unable to allocate Output file outdsn Return Code = RC

Explanation: The ALLOC command failed for file outdsn.

Operator Response: RC is the return code from the ALLOC command. Use this to determine why the allocate command failed, resolve the problem, and then retry the command.

Unable to allocate MIB Data file mibdata Return Code = RC

Explanation: The ALLOC command failed for file mibdata.

Operator Response: RC is the return code from the ALLOC command. Use this to determine why the allocate command failed, resolve the problem, and then retry the command.

Unable to Read the MIB Data file, Received Return Code = RC

Explanation: The REXX EXECIO function failed with return code RC for the MIB data file.

Operator Response: Use the RC to determine why the EXECIO function failed. Correct the problem and retry the command.

Unable to Write the Output file, Received Return Code = RC

Explanation: The REXX EXECIO function failed with return code RC for the Output file.

Operator Response: Use the RC to determine why the EXECIO failed. Correct the problem and retry the command.

No valid records created for this RFC File

Explanation: FKXECNVT was unable to understand any objects in the input RFC file, so no converted records were created.

Operator Response: Review the format of the RFC, and ensure it complies with the format restrictions defined in this book.
TCP/IP Command Support (IPCMD)

Syntax
Use the IPCMD routine to issue any TSO or UNIX command.

```
IPCMD
```

```
PROCEDURE=CNMSTYLE.TCPNAME
```

```
procedure=proc_name
```

```
SERVER=UNIX
```

```
SERVER=server_name
```

```
CMD=command_parms
```

Parameters

- **procedure**
  Specifies the MVS procedure. Valid values are PROC and STACK.
  - **proc_name**
    Specifies the name of the MVS procedure. The default is defined in sample CNMSTYLE with the common global variable TCPNAME. The value for TCPNAME is the `proc_name` of the TCP/IP stack. Refer to CNMSTYLE for more information about setting this value.

- **SERVER**
  Specifies the server name, either TSO or UNIX. The default is .

- **CMD**
  Specifies the TCP/IP command and parameters.

Usage

- Specify NETVASIS when entering UNIX commands or commands that are case sensitive.
- Parameters are not positional, but enter CMD=`command_parms` last to enable proper parameter parsing.
- The default stack and server can be changed by modifying CNMSTYLE. Refer to the comments in CNMSTYLE for more information.

Example

To use the UNIX server to ping host name quigley, using the default TCP/IP for MVS stack name, enter the following command:

```
netvasis IPCMD CMD=ping quigley
```

The response is returned to the caller.

To use the first available TSO server to ping host name quigley, enter the following command:

```
netvasis IPCMD SERVER=TSO CMD=ping quigley
```

The response is returned to the caller.
Programmatic Interface for IP Trace

Syntax

IP Trace does not have a command line interface. IP Trace uses an application programming interface and a series of common global variables that need to be set before invoking the IP Trace program. The following section provides information about these global variables. Refer to sample FHXETSMMP in the Tivoli NetView for z/OS Automated Operations Network User’s Guide for additional information.

To use IP Trace, the following required common global variables must be set.

For both Packet and Component Trace

FKXTRACE.sp.TARGET

sp The service point or stack for which these global variables are set.

target The target system and stack where the trace commands are sent. The target is a stemmed variable, which consists of the NetView domain and TCP/IP procedure name of the stack, in the format DOMAIN.PROC for the stack defined by sp.

For Component Trace

FKXTRACE.sp.CTRACE.OPTS1
FKXTRACE.sp.CTRACE.OPTS2
FKXTRACE.sp.CTRACE.IPADD1
FKXTRACE.sp.CTRACE.IPADD2
FKXTRACE.sp.CTRACE.IPADD3
FKXTRACE.sp.CTRACE.IPADD4
FKXTRACE.sp.CTRACE.PORT
FKXTRACE.sp.CTRACE.JOBS
FKXTRACE.sp.CTRACE.ASIDS

sp The service point or stack for which these global variables are set.

OPTS1 A list of CTRACE options. You can enter up to 20 options, separated by commas.

OPTS2 A list of up to 20 additional component trace options.

IPADDR1 A list of IP addresses. You can enter up to four addresses.

IPADDR2 A list of up to four additional IP addresses.

IPADDR3 A list of up to four additional IP addresses.

IPADDR4 A list of up to four additional IP addresses.

PORT A list of up to 16 IP ports, in the range 0–65535.

JOBS A list of up to 16 MVS job names, from 1–8 characters.

ASIDS A list of up to 16 MVS address space identifiers, in the form of four hexadecimal numbers.

For Packet Trace

FKXTRACE.sp.PKT.LINKS
FKXTRACE.sp.PKT.OPT.n
FKXETRA1

This global variable contains a total count of optional globals defined for this service point. For example, if there are six FXXTRACE.sp.PKT.OPT globals defined, the value of FXXTRACE.sp.PKT.LINKS is 6. Any number of optional globals can be set for all links on a service point. Each of these globals contains the following format:

LinkName,Len,Proto,Ipadd,Subnet,DestPort,SourcePort

**Note:** Use one global for each service point. When using the 3270 interface, all of these global variables are managed automatically.

**LinkName**
The TCP/IP device name on the service point traced.

**Len**
Specifies that a truncated portion of the IP packet can be traced. You can specify a length in the range 1–65535. The maximum value is FULL, which captures the entire packet.

**Proto**
The protocol collecting data. Valid values are:
- Asterisk (*), which specifies that packets of any protocol are traced
- ICMP
- RAW
- TCP
- UDP
- number (in the range 0–255)

**Ipadd**
The IP address that is compared with both the source and destination addresses of inbound and outbound packets. If either the source or destination address of a packet matches the specified IP address, the packet is traced. The IP address is specified in dotted decimal notation. If Ipadd is blank or an asterisk (*) is specified, all IP addresses are traced.

**Subnet**
The subnet mask that applies to the host and network portions of the IP address specified on the corresponding Ipadd.

**DestPort**
The port number that is compared with the destination port of inbound and outbound packets. The port number is an integer in the range 1–65535. If the destination port of a packet is the same as the specified port number, the packet is traced. This comparison is only performed for packets using either the TCP or UDP protocol. Packets using other protocols are not traced. If DestPort is omitted, there is no checking of the destination port of packets. If an asterisk (*) is specified, packets of any protocol and any destination port are traced.

**SourcePort**
The port number that is compared with the source port of inbound and outbound packets. The port number is an integer in the range 1–65535. If the source port of a packet is the same as the specified port number, the packet is traced. This comparison is only performed for packets using either the TCP or UDP protocol. Packets using other protocols are not traced. If SourcePort is omitted, there is no checking of the source port of packets. If an asterisk (*) is specified, packets of any protocol and any source port are traced.

**Syntax of FXXETRA1 Program**
The format consists of the following:

FKXETRA1 sp,type,action,writer,chron_start_date_time,duration,operator

*sp* The service point ID. This variable directs FXXETRA1 to use global variables to properly set up the trace.
**Example**

To start Packet Tracing for NMPIPL10, we need to setup the following common global variables:

```
FKXTRACE.NMPIPL10.PKT.LINKS = 3
FKXTRACE.NMPIPL10.PKT.OPT.1 = LOOPBACK,FULL,*,*,255.255.255.255,*,*
FKXTRACE.NMPIPL10.PKT.OPT.2 = TCPIPLINK,FULL,*,*,255.255.255.255,*,*
FKXTRACE.NMPIPL10.PKT.OPT.3 = TCPIPLINK000000,FULL,*,*,255.255.255.255,*,*
FKXTRACE.NMPIPL10.TARGET = LOCAL.TCPIP
```

To start a packet trace on this system, now, code the following within a REXX Clist:

```rexx
/*SAMPLE CODE TO START PKT TRACE NOW */
'FKXETRA1 NMPIPL10,PKT,START,PKTWRITER'
SAY 'PKT TRACE REQUEST ENDED WITH RC' RC
/* DISPLAY FXX MESSAGE, TOO */
'GLOBALV GETT EXLMSGTXT'*/
SAY EZLMSGTXT
EXIT
```
SNA Resource Automation (FKVESYNC)

Syntax

Quite often resources fail while NetView (or AON/SNA) is down. Because AON was not notified of the failure, AON automation routines do not request to recover the failed resources.

FKVESYNC is provided to assist you with automating your SNA resources. You can use FKVESYNC to determine which resources are down, and then drive AON automation routines to recover the failed resources.

FKVESYNC

Parameters

Defines which resource types to automate.

**ALL**

Automates all resources regardless of status. This is the default.

**APPLS**

Automates only the applications.

**CDRMS**

Automates only the cross-domain resource managers.

**CDRSCS**

Automates only the cross-domain resources.

**PHYSICAL**

Automates only the physical components. This includes NCPs, LUs, PUs, and switched nodes.

**SCOPE**

Defines the extent to which the resource is automated.

**EXCEPT**

Automates all resources that are currently not in an ACTIVE status.

This is the default.

**ALL**

Automates all resources regardless of status.

status

Automates only resources with this status.

~status

Automate all resources that do not have this status.
CHKAUTO
Determine if the resources displayed should be checked against the recovery statements in the control file when calling the NETSTAT command. The default is N.

DDF
Determine if the information should be passed to the Dynamic Display Facility (DDF) when calling the NETSTAT command. The default is N.

Usage

- FKVESYNC should be called from operator written code that is driven when NetView is initialized.
- FKVESYNC uses the AON NETSTAT command to determine the status of your SNA resources. In large networks with many failed resources, this process can take some time and CPU cycles.

Example

To check the status of all CDRMs, enter the following command:

FKVESYNC CDRMS

CNM377I FKVESYNC : INPUT ACCEPTED AND BEING PROCESSED ...
PLEASE WAIT
EZL001I REQUEST NETSTAT WAS SUCCESSFUL FOR OPER4

This detects all of your failed CDRM resources and drives AON automation routines to attempt the recovery of each. When you see message EZL001I, then AON recovery has been driven for each failed resource.
Chapter 9. Tailoring Gateways and Focal Points

Gateways are inbound and outbound automated operators. Focal points are domains to which all notifications are forwarded. Gateways and focal points enable you to define the following:

- Focal point host
- Backup focal point host
- Intermediate host or hosts
- Distributed host or hosts
- Adjacent host environment or environments

The message/alert router forwards messages and alerts from multiple hosts to a single host. This enables a network operator to receive all the network alert messages at a single console. The message destination is controlled within control file entries. Routed messages contain host origination identification.

To enable notification forwarding for AON you must complete the following steps:

- Determine the notification forwarding hierarchy.
- Tailor the NetView definitions.
- Define the focal point and backup focal point entries.
- Define the outbound gateway operator entries.
- Add NetView outbound and inbound operator IDs.
- Implement the changes in a host-by-host approach.

Refer to the example following each step for more detail.

1. Determine the notification forwarding hierarchy.
   
   You must determine the relationship between the different systems in a notification forwarding network. You must designate a primary focal point where you can send the messages. You can designate a backup for the primary focal point for situations when the primary focal point is unavailable. You must define the connectivity between the hosts in a tree-structured hierarchy, so that all notification messages are forwarded up to the primary focal point.
   
   You should develop a chart depicting the notification forwarding hierarchy. Figure 28 on page 206 shows an example of a notification forwarding chart.
In this example, CNM01 is the primary focal point and CNM99 is the backup focal point. The backup focal point does not receive notifications unless the primary focal point is down. As soon as the primary focal point is operational and the gateway connections are reestablished, the backup stops receiving notifications.

2. Tailor the NetView definitions.

**Note:** It is assumed that you have implemented all VTAM definitions required to permit cross-domain logons to remote NetView programs.

Add an RRD statement (if one does not already exist) in the CNMSTYLE member for each host with which this host directly communicates. Adding this NetView definition in CNMSTYLE member supports notification forwarding.

3. Define the focal point and backup focal point entries.

The focal point control file entry identifies the NetView domain where you forward the notification messages. Each focal point control file entry defines the final destination. Add a focal point definition control file entry to each distributed host.

An example of a focal point definition entry is:
4. Define the outbound gateway operator entries.

The outbound gateway operator establishes and maintains all of the outbound
connections to the domains defined in the GATEWAY control file entries. The
outbound gateway operator is defined in the AUTOOPS entry of the control file
as GATOPER.

AUTOOPS GATOPER,ID=name

The GATEWAY entries define the other domains with which a session should
be established so that notification messages can be forwarded to or from these
domains.

GATEWAY CNM02,DESC='NETVIEW CNM02',PASSWORD=password

NetView-NetView (NNT) sessions are established to all the domains defined by
attempting to log on to the inbound gateway operator of each domain. Each
inbound gateway operator has the same operator ID as the sending outbound
gateway operator.

5. Define NetView outbound and inbound operator IDs to the DSIOPF profile.

Define the inbound and outbound operator IDs to support notification
forwarding. A sample operator ID (GATCNM01) is provided.

Define one outbound operator ID. The outbound operator ID is the ID defined
in the AUTOOPS GATOPER control file entry in your domain.

Define one inbound operator ID for each domain defined in a GATEWAY
control file entry. For the inbound operator ID, use the ID defined in the
AUTOOPS GATOPER control file entry in the other domains.

Note: You must also specify the logon ID in the GATEWAY statement of the
remote domain.

6. Implement the changes in one NetView host at a time.

The notification forwarding implementation is best approached as a top-down
implementation starting with the primary focal point, then the distributed
hosts. This approach works best because the focal point is ready to handle the
forwarded notifications when notification forwarding is enabled in the remote
hosts.

When using a top-down approach, if your installation has not implemented
message forwarding, the messages at the remote hosts are displayed to
notification operators at the remote hosts. After message forwarding is turned
on, the messages are routed to the focal point and appropriately handled. Local
notification operators, if any remain, are also notified appropriately.

---

AON Focal Point Compatibility

The following restrictions apply to the AON-supported focal point compatibility:

- AON does not support intermediate focal point domains. You can use
  full-function domains as intermediate routing domains, but AON does not act
  on data passing through an intermediate routing domain, and does not update
  DDF in the intermediate domain.

- Down-level distributed domains must be directly connected to the focal point
domain. You cannot use any ADJNETV statements.

- Down-level domains might not act as an intermediate. You cannot specify them
  in the DOMAIN or ALTNETV parameters on any ADJNETV statement.
- Down-level focal point domains must have all distributed domains directly connected. You cannot use any ADJNETV statements.
- Down-level compatibility is provided to ease migration to current levels of gateway support. Down-level compatibility is not intended for long-term production operation. Migration of all domains to full-function support should be accomplished as quickly as possible.

Table 10 illustrates allowable communications setups. Wherever a down-level domain is connected to a full-function domain, the full-function domain should be AON. Wherever two full-function domains are connected, the full-function domains can be any supported full-function product.

**Table 10. Communications Setup in Focal Point Domain Environment**

<table>
<thead>
<tr>
<th>Focal point support</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. DL(dn) → AON(fp)</td>
</tr>
<tr>
<td>2. AON(dn) → DL(fp) (except AON SolutionPac®)</td>
</tr>
<tr>
<td>3. FF(dn) → FF(fp)</td>
</tr>
<tr>
<td>4. FF(dn) → FF(ir) → FF(fp)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SENDCMD support</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. DL ↔ AON</td>
</tr>
<tr>
<td>2. AON ↔ DL</td>
</tr>
<tr>
<td>3. FF ↔ FF</td>
</tr>
<tr>
<td>4. FF ↔ FF ↔ FF</td>
</tr>
<tr>
<td>5. DL ↔ AON ↔ DL</td>
</tr>
<tr>
<td>6. DL ↔ AON ↔ FF</td>
</tr>
<tr>
<td>7. FF ↔ AON ↔ DL</td>
</tr>
</tbody>
</table>

**Notes:**
1. The abbreviations in this table have the following meanings:
   - AON: Full-function domain (AON)
   - FF: Full-function domain
   - DL: Any down-level domain
   - (dn): Distributed domain
   - (fp): Focal point domain
   - (ir): Intermediate routing domain
2. SENDCMD support for these is the same.
3. Not valid for focal point support.

**Figure 29** illustrates a typical migration environment:

**Figure 29. Typical Migration Environment**

Table 11 on page 209 illustrates the focal-point and SENDCMD support for a typical migration environment.
Table 11. Focal-point and SENDCMD Support for Typical Migration Environment

<table>
<thead>
<tr>
<th>Support type</th>
<th>Between domains</th>
<th>Communications setup</th>
</tr>
</thead>
<tbody>
<tr>
<td>Focal point</td>
<td>CNM02 to CNM01</td>
<td>DL(dn) → FF(fp)</td>
</tr>
<tr>
<td></td>
<td>CNM03 to CNM01</td>
<td>DL(dn) → FF(fp)</td>
</tr>
<tr>
<td></td>
<td>CNM04 to CNM01</td>
<td>FF(dn) → FF(fp)</td>
</tr>
<tr>
<td>SENDCMD</td>
<td>CNM02 to CNM01</td>
<td>DL ↔ FF</td>
</tr>
<tr>
<td></td>
<td>CNM01 to CNM03</td>
<td>FF ↔ DL</td>
</tr>
<tr>
<td></td>
<td>CNM04 to CNM01</td>
<td>FF ↔ FF</td>
</tr>
<tr>
<td></td>
<td>CNM02 to CNM03 (through CNM01)</td>
<td>DL ↔ FF ↔ DL</td>
</tr>
<tr>
<td></td>
<td>CNM02 to CNM04 (through CNM01)</td>
<td>DL ↔ FF ↔ FF</td>
</tr>
<tr>
<td></td>
<td>CNM04 to CNM03 (through CNM01)</td>
<td>FF ↔ FF ↔ DL</td>
</tr>
</tbody>
</table>

Figure 30 shows one possible example of a full-function environment.

Table 12 lists focal-point and SENDCMD support for the full-function environment as previously illustrated.

Table 12. Focal-point and SENDCMD Support for Full-function Environment

<table>
<thead>
<tr>
<th>Support type</th>
<th>Between domains</th>
<th>Communications setup</th>
</tr>
</thead>
<tbody>
<tr>
<td>Focal point</td>
<td>CNM51 to CNM50</td>
<td>FF(dn) → FF(fp)</td>
</tr>
<tr>
<td></td>
<td>CNM52 to CNM50 through CNM51</td>
<td>FF(dn) → FF(ir) → FF(fp)</td>
</tr>
<tr>
<td>SENDCMD</td>
<td>CNM51 to CNM50</td>
<td>FF ↔ FF</td>
</tr>
<tr>
<td></td>
<td>CNM52 to CNM50 through CNM01</td>
<td>FF ↔ FF ↔ FF</td>
</tr>
</tbody>
</table>

Sample definitions for this full-function environment for each domain follow:

- For domain CNM50:
Notification Forwarding Example

Assume that the AON operator AUTMON wants to forward a request (AON notify) to CNM02 from CNM01. The outbound gateway operator for CNM01 is GATCNM01. The outbound gateway operator for CNM02 is GATCNM02. The inbound gateway operator for CNM02 that receives notification messages from CNM01 is GATCNM01.

The program sends the request to the CNM01 outbound gateway operator (GATCNM01). The outbound gateway operator determines if the request is for the CNM01 domain. In this example, it is not. The outbound gateway operator routes the request to the inbound gateway operator in domain CNM02 (GATCNM01) from domain CNM01. When the inbound gateway operator receives the request, it is sent to the outbound gateway operator for CNM02 (GATCNM02). The outbound gateway operator from CNM02 determines that the forwarded request is for this domain and issues the request. Refer to Figure 31.

Table 13 on page 211 lists the required control file entries for the domains illustrated in Figure 31.
<table>
<thead>
<tr>
<th>Domain</th>
<th>Control file entry</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>CNM01</td>
<td>AUTOOPS</td>
<td>AUTOOPS GATOPER,ID=GATCNM01</td>
</tr>
<tr>
<td></td>
<td>Automation operator</td>
<td></td>
</tr>
<tr>
<td></td>
<td>GATEWAY</td>
<td>GATEWAY CNM02,DESC='NEXT DOMAIN', PASSWORD=pswd</td>
</tr>
<tr>
<td></td>
<td>Forwarded domains and operator definitions</td>
<td></td>
</tr>
<tr>
<td></td>
<td>FORWARD FOCALPT</td>
<td>FORWARD FOCALPT,PRI=CNM99</td>
</tr>
<tr>
<td></td>
<td>Focal point definitions</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ADJNETV</td>
<td>ADJNETV CNM99,DOMAIN=CNM02</td>
</tr>
<tr>
<td></td>
<td>Focal point definitions</td>
<td></td>
</tr>
<tr>
<td>CNM02</td>
<td>AUTOOPS</td>
<td>AUTOOPS GATOPER,ID=GATCNM02</td>
</tr>
<tr>
<td></td>
<td>Automation operator</td>
<td></td>
</tr>
<tr>
<td></td>
<td>GATEWAY</td>
<td>GATEWAY CNM99,DESC='NEXT DOMAIN', PASSWORD=pswd</td>
</tr>
<tr>
<td></td>
<td>Forwarded domains and operator definitions</td>
<td></td>
</tr>
<tr>
<td></td>
<td>GATEWAY</td>
<td>GATEWAY CNM01,DESC='DOWNSTREAM', PASSWORD=pswd</td>
</tr>
<tr>
<td></td>
<td>FORWARD FOCALPT</td>
<td>FORWARD FOCALPT,PRI=CNM99</td>
</tr>
<tr>
<td></td>
<td>Focal point definitions</td>
<td></td>
</tr>
<tr>
<td>CNM99</td>
<td>AUTOOPS</td>
<td>AUTOOPS GATOPER,ID=GATCNM99</td>
</tr>
<tr>
<td></td>
<td>Automation operator</td>
<td></td>
</tr>
<tr>
<td></td>
<td>GATEWAY</td>
<td>GATEWAY CNM02,DESC='DOWNSTREAM', PASSWORD=pswd</td>
</tr>
<tr>
<td></td>
<td>Forwarded domains and operator definitions</td>
<td></td>
</tr>
<tr>
<td></td>
<td>FORWARD FOCALPT</td>
<td>FORWARD FOCALPT,PRI=CNM99</td>
</tr>
<tr>
<td></td>
<td>Focal point definitions</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ADJNETV</td>
<td>ADJNETV CNM01,DOMAIN=CNM02</td>
</tr>
<tr>
<td></td>
<td>Focal point definitions</td>
<td></td>
</tr>
</tbody>
</table>
Chapter 10. AON User Exits

With AON you can write user exits to tailor automation for a specific task. An exit can call any command, program, command list, or EXEC that can be issued from a NetView command line. Exits are processed by an automation operator, so processing restrictions that apply to automation operator-run programs also apply to exit programs. Refer to *Tivoli NetView for z/OS Customization: Using REXX and the NetView Command List Language* for information about these restrictions.

User exits alter task global variables or return codes set by some common routines. These common routines guide further automation notification and logging or initiate activity based on failure or other conditions defined to AON. You can write exits that modify the task global variables and return codes when the following AON functions are issued:

- SNA resource information gathering (EZLEAGRN)
- Thresholds checking (EZLEATHR)
- Automation flag analysis based on recovery statements (EZLECAUT)
- Resource recovery attempts on MONIT intervals (EZLECATV)
- AON message issued to logs or operators
- NCP recovery

Exits invoked by common routines can be called with input parameters. Valid input parameters for exit routines include the task global variables set by the common routine, user-defined literals, and the resource name sent to the common routine.

Exits to be called are defined in the control file. Default exits are coded on the ENVIRON EXIT entry. These exits are called every time the common routine is invoked. Table 14 identifies which routine calls which exit.

### Table 14. Common Routines That Call Exits

<table>
<thead>
<tr>
<th>Common routine</th>
<th>Policy entry that sets default</th>
<th>Control file entry that sets specific call</th>
</tr>
</thead>
<tbody>
<tr>
<td>EZLEAGRN</td>
<td>ENVIRON EXIT (EXIT05)*</td>
<td>n/a</td>
</tr>
<tr>
<td>EZLEATHR</td>
<td>ENVIRON EXIT (EXIT06)*</td>
<td>THRESHOLDS</td>
</tr>
<tr>
<td>EZLECAUT</td>
<td>ENVIRON EXIT (EXIT07)*</td>
<td>RECOVERY</td>
</tr>
<tr>
<td>AON messaging</td>
<td>ENVIRON EXIT (EXIT08)*</td>
<td>n/a</td>
</tr>
<tr>
<td>EZLECATV</td>
<td>ENVIRON EXIT (EXIT09)*</td>
<td>MONIT</td>
</tr>
<tr>
<td>EZLENTFY</td>
<td>n/a</td>
<td>NOTIFY EXIT (EXIT10)</td>
</tr>
<tr>
<td>EZLENFRM</td>
<td>SETUP EXIT (EXIT11)+</td>
<td>n/a</td>
</tr>
<tr>
<td>EZLENFRM</td>
<td>SETUP EXIT (EXIT12)+</td>
<td>n/a</td>
</tr>
</tbody>
</table>

**Note:** Specific entries override the default entries.

* Indicates that the entry is defined in the control file.

+ Indicates that the entry is defined in the Inform Policy member.

You can also define different exits on the control file entry applicable to a particular resource or class of resources for each of the exit points (by using wildcard characters or resource types in definitions). These are defined on the RECOVERY, THRESHOLDS, and MONIT entries for the resource. You can specify
up to 10 exits for each common routine. If you specify more than one exit routine, each exit will be called until the first nonzero return code is received from the user-written exit routine. When multiple exits are called, AON uses the task global variable values set by the series of exits, up to and including the last exit issued. Table 15 identifies the task global variable values set by exits.

Table 15. Task Global Variables Set by Exits

<table>
<thead>
<tr>
<th>Common Routine</th>
<th>Exit Number</th>
<th>Policy Entry</th>
<th>Task global variable</th>
<th>Exit task global variable</th>
<th>Issued</th>
</tr>
</thead>
<tbody>
<tr>
<td>EZLEAGRN</td>
<td>EXIT05</td>
<td>n/a</td>
<td>RESNAME</td>
<td>EZLEAGRNRC</td>
<td>Before EZLEAGRN returns control to the calling program.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>RESTYPE</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>RESSTAT</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>RESMAJ</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>RESLINE</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>RESNODE</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>RESPU</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>RESSA</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>RESSW</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EZLEATHR</td>
<td>EXIT06</td>
<td>THRESHOLDS*</td>
<td>RESNAME</td>
<td>EZLEATHRRC</td>
<td>After the type of threshold exceeded is determined.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>RESTYPE</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>RESSTAT</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>RESMAJ</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>RESLINE</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>RESNODE</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>RESPU</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>RESSA</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>RESSW</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EZLECAUT</td>
<td>EXIT07</td>
<td>RECOVERY*</td>
<td>RESNAME</td>
<td>EZLECAUTRC</td>
<td>Each time the recovery flag is checked for a resource.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>RESTYPE</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>RESSTAT</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>RESMAJ</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>RESLINE</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>RESNODE</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>RESPU</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>RESSA</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>RESSW</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AON Messaging</td>
<td>EXIT08</td>
<td>n/a</td>
<td>EZLCONVERT</td>
<td>n/a</td>
<td>When issuing an AON message.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>EZLMSGTXT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EZLECATV</td>
<td>EXIT09</td>
<td>MONIT*</td>
<td>RESNAME</td>
<td>EZLECATVRC</td>
<td>Prior to deactivation of a resource.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>RESTYPE</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>RESSTAT</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>RESMAJ</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>RESLINE</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>RESNODE</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>RESPU</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>RESSA</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>RESSW</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EZLENTFY</td>
<td>EXIT10</td>
<td>NOTIFY*</td>
<td>EZLNTFYLST</td>
<td>n/a</td>
<td>Prior to any notification actions.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>EZLMSGTXT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EZLENFRM</td>
<td>EXIT11</td>
<td>SETUP+</td>
<td>EZLPOLEX</td>
<td>n/a</td>
<td>Prior to consulting the inform policy.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>EZLPOLEX</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EZLENFRM</td>
<td>EXIT12</td>
<td>SETUP+</td>
<td>n/a</td>
<td></td>
<td>Prior to calling the inform interface routine.</td>
</tr>
</tbody>
</table>

**Attention:** The task global variables contain data that guide AON recovery processing. Indiscriminate changes to the task global variables can cause unpredictable results.

* Indicates that the entry is defined in the control file.

+ Indicates that the entry is defined in the Inform Policy member.
During initialization common global variables are set for the default exits for each exit point. If the ENVIRON EXIT control file entry is modified, reload the control file. The new exit common global variable values will be changed. Exit definitions on the THRESHOLDS, RECOVERY, or MONIT entries take effect immediately and do not require the control file to be reloaded when they are updated online.
With NCP recovery, you can specify user exits to run during different stages. Four separate user exit statements are used for each of the four stages where an exit can be used.

**EXIT01**  
User-defined command or command list. The program defined in EXIT01 runs after the dump message response. Sample exit routines are provided (FKVEX01 in CNMSAMP).

**EXIT02**  
User-defined command or command list. The program defined in the EXIT02 exit runs after the dump is complete and the dump time exceeded timer is purged. Sample exit routines are provided (FKVEX02 in CNMSAMP).

**EXIT03**  
User-defined command or command list. The program defined in the EXIT03 exit runs after the reload message response. Sample exit routines are provided (FKVEX03 in CNMSAMP).

**EXIT04**  
User-defined command or command list. The program defined in the EXIT04 exit runs after the load is complete and the load time exceeded timer is purged. Sample exit routines are provided (FKVEX04 in CNMSAMP).

**Syntax**

```plaintext
NCPRECOV

ncpname, HOST=domain [EXIT01=cmd] [EXIT02=cmd] [EXIT03=cmd] [EXIT04=cmd]
```

*ncpname*  
Specifies the NCP for which these exits are being used.

*command*  
Specifies an exit to be run or a command to be issued.

**Usage**

To use these exits, copy them into your user library from the DSICLD data sets.

If response to the option to dump the NCP is specified as N, then EXIT02 is not run because no message is received, thus indicating that a dump is complete.

If response to the option to load the NCP is specified as N, then EXIT04 is not run because no message is received, thus indicating that a load is complete.

Variable parameters cannot be passed for these exits. For example, USERPGM &NCP is not valid.

**Attention:** The task global variable (TGLOBALs) contain data that guide AON/SNA recovery processing. Indiscriminate changes to these task global variables (TGLOBALs) can cause unpredictable results.
Example

In the following example, AON runs the FKVEX01 exit when it replies to the request to dump message. AON runs the FKVEX02 exit when the dump is complete. AON runs the FKVEX03 exit when it responds to the reloading of the NCP. AON runs the FKVEX04 exit when it completes the reloading of the NCP.

NCPRECOV NCPABC,
   HOST=CNM01,DUMP=(Y,N),RELOAD=(Y,N),
   LINKSTA=123-S,DUMPTSTA=123-S,
   LOADTIME=00:15,DUMPTIME=00:10,
   EXIT01=FKVEX01,
   EXIT02=FKVEX02,
   EXIT03=FKVEX03,
   EXIT04=FKVEX04
EXIT05 Processing (EZLEAGRN)

Purpose
Exit processing for EZLEAGRN is issued just before EZLEAGRN returns control to the calling program. AON gathers all information about a resource from a VTAM DISPLAY NET command and sets the following task global variables:

- **RESTYPE**  Resource type
- **RESSTAT**  Resource status
- **RESMAJ**   The major node of the resource
- **RESLINE**  The higher node line of the resource
- **RESNODE**  The adjacent major node of the resource
- **RESPU**    The higher node controller of the resource
- **RESSA**    The subarea of the resource
- **RESSW**    The switched major node of the resource

Any of these task global variables can be altered. The return code passed to the calling program by EZLEAGRN can be set from a user exit. The EZLEAGRN return code does not guide automation activity in AON, but the attribute can be adjusted by the exit. The altered return code is set by the user exit and placed in the task global variable, EZLEAGRNRC.

EZLEAGRN exits can be defined only in the control file on the ENVIRON EXIT by the EXIT05 parameter. The previously mentioned task global variables can be passed to the exit by specifying the task global variable name preceded by an ampersand (&). The resource name can also be passed to the exit routine by specifying &RESNAME.

Syntax
ENVIRON EXIT

Parameters
- **command**  Specifies the exit or command to be issued.
- **parms**    Specifies the parameters to be passed to the exit or the command to be issued.

Usage
A sample exit is provided in CNMSAMP member FKVEXT05.

Run all exits inline. Do not route exits to another task.

Code user exits to change any of the task global variables set by EZLEAGRN. These task global variables are set from the results of a VTAM DISPLAY command for the resource.
EXIT05 (EZLEAGRN)

If any user-coded routines are written in REXX and a task global variable is changed, retrieve the new value by using the REXX ‘GLOBALV GETT’ statement.

Set task global variable EZLEAGRNRC in the user exit to alter the return code for EZLEAGRN.

All defined user exits will be called in the order they are coded in the control file. All exits will be issued until the user exit returns a nonzero return code. This return code is different from the return code set in task global variable EZLEAGRNRC.

EXIT05 can be used to change resource data used by AON or set additional task global variables from running other commands or message processing. For example, you could set up additional resource types to be used to guide recovery flags and DDF logging by analyzing the resource name and changing the RESTYPE task global variable.

Attention: The task global variables contain data that guide AON recovery processing. Indiscriminate changes to the task global variables can cause unpredictable results.

Example

For this example, each time EZLEAGRN is issued, FKVEXT05 is issued and passed the value of RESNAME as the first parameter, the literal TEST as the second parameter, and the value of task global variable RESTYPE as the third parameter.

ENVIRON EXIT,EXIT05=(*FKVEXT05 &RESNAME TEST &RESTYPE*)
EXIT06 Processing (EZLEATHR)

Purpose
Exit processing for EZLEATHR is issued after EZLEATHR performs threshold analysis and before operator notifications have been issued. The resource name and type must be passed to EZLEATHR. AON sets the following task global variables in EZLEATHR:

- **RESTYPE**: Resource type
- **EZLTRSHLD**: Threshold exceeded value (NONE, CRIT, FREQ, INFR)
- **EZLRECORD**: Specifies whether this error is to be saved in the status file (Y,N)
- **EZLNOTIFY**: Specifies whether any operator notifications are to take place (Y,N)
- **EZLCRIT**: Indicates whether a critical threshold has been exceeded (Y,N)
- **EZLFREQ**: Indicates whether a frequent threshold has been exceeded (Y,N)
- **EZLINFR**: Indicates whether an infrequent threshold has been exceeded (Y,N)

The return code passed to the calling program by EZLEATHR can be set from EXIT06. The altered return code is set by EXIT06 and placed in the task global variable EZLEATHRRRC. The return codes from EZLEATHR are used by calling programs to determine which threshold has been exceeded and if automation should continue.

Table 16. EZLEATHR results

<table>
<thead>
<tr>
<th>If EZLEATHR determines...</th>
<th>...then....</th>
</tr>
</thead>
<tbody>
<tr>
<td>a critical threshold has been exceeded</td>
<td>a return code of 3 is passed back to the calling program.</td>
</tr>
<tr>
<td>a frequent threshold has been exceeded</td>
<td>a return code of 2 is passed back to the calling program.</td>
</tr>
<tr>
<td>an infrequent threshold has been exceeded</td>
<td>a return code of 1 is passed back to the calling program.</td>
</tr>
</tbody>
</table>

EZLEATHR exits can be defined in the control file on the ENVIRON EXIT by the EXIT06 parameter or on the THRESHOLDS control file entry. Task global variables can be passed to the exit by specifying the task global variable name preceded by an ampersand (&). Resource names may also be passed to the exit routine by specifying &RESNAME.

**Syntax**

**ENVIRON EXIT**

```
ENVIRON EXIT
```

THRESHOLDS entry for the specific resource:
Parameters

**command**
- Specifies the exit or command to be issued.

**parms**
- Specifies the parameters to be passed to the exit or command to be issued.

Usage

A sample exit is provided in CNMSAMP member EZLEXT06.

Run all exits inline. Do not route exits to another task.

Code user exits to change any of the task global variables set by EZLEATHHR.

If any user-coded routines are written in REXX and a task global variable is changed, retrieve the new value by the use of the REXX ‘GLOBALV GETT’ statement.

Set task global variable EZLEATHRRC in the user exit to alter the return code for EZLEATHHR.

If an exit parameter is coded on the THRESHOLDS entry, the exits coded on the THRESHOLDS entry will be issued.

All defined user exits will be called in the order they are coded in the control file. All exits will be issued until the user exit returns a non-zero return code. This return code is different from the return code set in task global variable EZLEATHRRC.

If the threshold exceeded type is altered, make the change in the EZLTRSHLD task global variable.

EXIT06 can be used to change the thresholding analysis results used by AON to guide automation and notifications based on thresholds exceeded. This might be desirable if there are no other factors influencing failure rates or resources which are not available to AON. EXIT06 could also perform some function based on thresholding exceptions, but not necessarily change the results of the analysis.

**Attention:** The task global variables contain data that guide AON recovery processing. Indiscriminate changes to the task global variables can cause unpredictable results.

Example

For this example, each time the THRESHOLDS FLAG is checked for resource ABPU, EZLEXT06 is issued and passed the value of the task global variable...
EXIT06 (EZLEATHR)

RESNAME as the first parameter, the value of task global variable RESTYPE as the second parameter, and the type of threshold that was exceeded as the third parameter:

THRESHOLDS ABPU,
    CRIT=(4,00:12),
    FREQ=(4,01:00),
    INFIR=(5,24:00),
EXIT06="(EZLEXT06 &RESNAME &RESTYPE &EZLTRSHLD")"
EXIT07 Processing (EZLECAUT)

Purpose
Exit processing for EZLECAUT is issued when AON checks the automation status of a resource. Automation status is set by analyzing the RECOVERY AUTO and NOAUTO parameters entries. However, if the EZLECAUT higher node of the resource is inactive, AON will not analyze the RECOVERY entries. In this case, EXIT07 will not be called. For example, if EZLECAUT is called with ABPU and the line ABLINE1 associated with ABPU is not in an active state, exit processing will not be issued for resource ABPU.

AON uses the EZLECAUT return code to determine whether automation is on or off for the resource. A return code greater than zero (0) indicates automation should not occur. A return code of zero (0) indicates automation is to proceed. The return code passed to the calling program by EZLECAUT can be set from a user exit. The altered return code is set by the user exit and placed in the task global variable EZLECAUTRC.

EZLECAUT exits can be defined in the control file on the ENVIRON EXIT by the EXIT07 parameter and the RECOVERY entry that applies to the resource. The previously mentioned task global variables can be passed to the exit by specifying the task global variable name preceded by an ampersand (&). The resource name may also be passed to the exit routine by specifying &RESNAME.

Syntax
ENVIRON EXIT

```plaintext
EXIT07="command parms"
```

RECOVERY

```plaintext
resource, AUTO=Ｙ
EXIT07="command parms"
```

Parameters
- **resource**: Specifies the resource name.
- **command**: Specifies the exit or command to be issued.
- **parms**: Specifies the parameters to be passed to the exit or the command to be issued.
EXIT07 (EZLECAUT)

Usage

A sample exit is provided in CNMSAMP member EZLEXT07.

Run all exits inline. Do not route exits to another task.

Code user exits to change task global variables set by EZLECAUT. These task global variables are set from the results of a VTAM DISPLAY command for the resource.

If any user-coded routines are written in REXX and a task global variable is changed, retrieve the new value by the use of the REXX 'GLOBALV GETT' statement.

Set task global variable EZLECAUTRC in the user exit to alter the return code for EZLECAUT.

If an exit parameter is coded on the RECOVERY entry, the exits coded on the RECOVERY entry will be issued.

All defined user exits will be called in the order they are coded in the control file. All exits are issued until the user exit returns a nonzero return code. This return code is different from the return code set in task global variable EZLECAUTRC.

A return code of zero from EZLECAUT indicates that recovery for this resource is to continue. Any other return code from EZLECAUT indicates that automation should stop.

If the parameters sent to EZLECAUT include the resource name, type and higher node name, EZLEAGRN is not called for the resource. If any of those parameters are missing, EZLEAGRN is issued to retrieve the resource type. As a result EXIT05 is issued for the resource. Standard AON processing sends only the resource name; therefore, each call to EZLECAUT and EXIT07 can cause an execution of EZLEAGRN and EXIT05.

You can use EXIT07 to influence the automation flag and AON recovery activity based on data not readily available to AON. For example, the EXIT checks a database, schedule, or issue commands to another equipment manufacturer (OEM) device to further determine whether resource recovery is appropriate. EXIT07 can also initiate some activities that do not change the automation flag; however, that should occur only because AON has detected an error condition.

Attention: The task global variables contain data that guide AON recovery processing. Indiscriminate changes to the task global variables can cause unpredictable results.

Example

For this example, each time the RECOVERY FLAG is checked for resource ABPU, EZLEXT07 is issued and passed the value of the task global variable RESNAME as the first parameter and the value of task global variable RESTYPE as the second parameter:

```
RECOVERY ABPU,
   AUTO=Y,
   EXIT07=("EZLEXT07 &RESNAME &RESTYPE")
```
EXIT08 Processing (AON Messaging)

Purpose

The AON messaging exit enables you to run your own program when an AON message is sent to operators. EXIT08 is defined on the ENVIRON EXIT entry and is specified by the EXIT08 parameter.

The AON messaging exit enables you to analyze the resource name and type and assign a new valid AON message class for this resource when an alert is to be sent to a focal point (either AON or the systems operations component of System Automation for OS/390).

The resource name, type, and any user defined parameters can be passed to the user exit defined in the EXIT08 parameter. The parameters are received by the exit in the order that they are passed. A sample exit has been provided in CNMSAMP (EZLEXT08). The assigned message class or classes are stored in the task global variable EZLCONVERT. If multiple message classes are assigned, the classes must be separated by blanks. Any return code set by the user exit will be ignored. The resource name and resource type can be passed to the exit by specifying the RESNAME and RESTYPE respectively.

You can also pass the message text using variable MSGTEXT,EZLMSGTXT or referring to the task global variable EZLMSGTXT in your program. Your program can then send the message to other operator notification devices like pagers or status boards. It could initiate a note in an electronic mail system. Most external notification devices have workstation interfaces that can be accessed from a service point. Some service points can be driven by NetView or MVS commands issued from NetView programs.

Syntax

ENVIRON EXIT

EXIT08={"command"""parms""")

Parameters

<table>
<thead>
<tr>
<th>command</th>
<th>Specifies the exit or command to be issued</th>
</tr>
</thead>
<tbody>
<tr>
<td>parms</td>
<td>Specifies parameters that are to be passed to the exit routine</td>
</tr>
</tbody>
</table>

Usage

A sample exit is provided in CNMSAMP member EZLEXT08.

Run all exits inline. Do not route exits to another task.

All return codes from the user exit will be ignored.

Return message classes in task global variable EZLCONVERT. Message classes are a maximum of two characters separated by blanks.
EXIT08 (Messaging)

Attention: The task global variables contain data that guides AON recovery processing. Indiscriminate changes to the task global variables can cause unpredictable results.

Example

In this example, EZLEXT08 is issued when a message is issued:

```plaintext
ENVIRON EXIT,EXIT08=('EZLEXT08 &RESNAME &RESTYPE')
```

EZLEXT08 is passed the values for resource name and resource type.
EXIT09 Processing (EZLECATV)

Purpose

Exit processing for EZLECATV is issued after the MONIT interval data is returned and before activation or deactivation of the resource occurs. You can set the following TGLOBALs to influence recovery in EXIT09:

- **EZLNXTINTVL**: Change this value to indicate whether MONIT intervals are to continue. The default is yes (Y).
- **EZLNOTIFY**: Change this value to specify a different notify flag for the current MONIT interval (can suppress or force operator reminder notification on this interval).
- **EZLSCOPE**: Change this value to specify the scope parameter for VTAM VARY ACT command. The default scope value is U for NCPs, lines, and PUs.
- **EZLACT**: Change this value to specify whether to attempt to activate the resource. Valid values are yes (Y) and no (N). The default is yes (Y).
- **EZLINACT**: Change this value to specify whether to attempt to deactivate the resource. Valid values are yes (Y) and no (N). The default is no (N) for resource types of LINKSTA or CDRSC, otherwise the default is yes (Y).
- **EZLMONIT**: This value cannot be changed but can be passed to the exit. This TGLOBAL contains the MONIT value that is currently being run.
- **EZLINTVL**: This value cannot be changed but can be passed to the exit. This TGLOBAL contains the number of the MONIT interval currently being processed.
- **RESTYPE**: Change this value to change the resource type of the resource.

The return code passed to the calling program by EZLECATV can be set from EXIT09. The altered return code is set by the user exit and placed in the TGLOBAL EZLECATVRC. The return code from EZLECATV is not used by AON for automation.

EZLECATV exits can be defined in the control file on the ENVIRON EXIT or the MONIT entries by the EXIT09 parameter. The resource name and resource type can be passed to the exit by specifying the &RESNAME and &RESTYPE respectively.
EXIT09 (EZLECATV)

Syntax

```
ENVIRON EXIT
```

Special exit routines can be coded for specific resources on the MONIT entry for the specific resource MONIT.

```
MONIT
```

Parameters

- **command**: Specifies the exit to be run or the command to be issued.
- **parms**: Specifies the parameters to be passed to the exit or the command to be issued.

Usage

A sample exit is provided in CNMSAMP member FKVEXT09.

Run all exits inline. Do not route exits to another task.

Code user exits to change the TGLOBALs set by EZLECATV.

If any user-coded routines are written in REXX and a TGLOBAL is changed, retrieve the new value by using the REXX 'GLOBALV GETT' statement.

Set TGLOBAL EZLECATVRC in the user exit to alter the return code for EZLECATV.

All defined user exits will be called in the order they are coded in the control file. All exits will be run until the user exit returns a non-zero return code. This return code is different from the return code set in TGLOBAL EZLECATVRC.

EXIT09 can be used to change the way AON recovers a resource, or class of resources by setting the TGLOBALs. EXIT09 can be used to perform some function which is desirable each time a recovery interval is processed, but not necessarily change the AON recovery logic.

**Attention**: The TGLOBALs contain data that guide AON recovery processing. Indiscriminate changes to the TGLOBALs can cause unpredictable results.
Example

In this example, on each MONIT interval for resource ABPU, FKVEXT09 will be run and passed the value of the TGLOBAL RESNAME as the first parameter, the value of TGLOBAL RESTYPE as the second parameter, the number that represents the MONIT interval cycle as the third parameter, and the MONIT value for this interval:

```
MONIT ABPU,
    INTVL=(00:02,Y),
    INTVL=(00:04,N),
    EXIT09=("FKVEXT09 &RESNAME &RESTYPE &EZLINTVL &EZLMONIT")
```
EXIT10 Processing (EZLENTFY)

**Purpose**

Exit processing for EZLENTFY is issued during AON notification processing. EZLENTFY builds a list of notification types which can be overridden in EXIT10. You can set the following task globals to influence how AON performs notification processing:

- **EZLNTFYLIST**
  This global variable contains a list of notification actions. The list can be one or more of the following:
  - ALERT
  - TEC
  - MSG
  - DDF
  - Inform_Policy_Name

- **EZLNTFYTYPE**
  This global variable contains the type of notification requested. This could be a resource name, resource type (for example PU), or event type (for example CRITTHRS).

**Syntax**

```
NOTIFY
  DEFAULTS
  ResType
  ResName
  Res*
  Event_Type
   ALERT= YES
   TEC= YES
   Inform= YES
   Inform_Policy_Name
   MSG= YES
   DDF= YES

EXIT10="command parms"
```

**Parameters**

- **DEFAULTS**
  Notification policy at a system wide, or defaults, level

- **ResType**
  Notification policy based on a type of resource

- **ResName**
  Notification policy based on a particular resource name

- **Res**
  Notification policy based on a range of resource names

- **Event_Type**
  Event_Type can be one of the following:
  - CRITTHRS
    Critical automation threshold exceeded
  - NOMOMONS
    No more monitoring intervals defined
REMIND
Reminder that a resource is still down

BRGCONGEST
Bridge Congestion Threshold exceeded

ADPCONGEST
Adapter Congestion Threshold exceeded

NAMESERV
NameServer Failure Threshold exceeded

ALERT
Specifies when to generate MSU notifications. MSUs are required for Tivoli Enterprise Console events.
YES Specifies to generate an MSU.
NO Specifies do not generate an MSU. No is also the default.
TEC Specifies to generate an MSU so that the Tivoli Enterprise Console will receive the notification.

INFORM
Specifies the use of pager/beeper or e-mail that is defined in the CONTACT entry statement of the Inform Policy. You can use different resources and resource types.

MSG
Specifies whether to generate a message or not.
YES Specifies to display a message.
NO Specifies not to display a message. No is the default.

DDF
Specifies how the DDF component is updated.
YES Log the event to DDF.
NO Do not log events to DDF. No is the default.

EXIT10
Is a user exit that enables the override of any of the notification actions.

command
Specifies the exit or command to be issued.

parms
Specifies the parameters that are to be passed to the exit routine.

Usage
It is possible for EXIT10 to be driven without any notification policy defined. In that case, EZLNTFYLST will be null. This would occur if the defined policy has been customized to not issue any notifications. The user exit can then enable particular notifications.

Example
In the following example, CLIST TEXIT10 will be called to override the existing notification policy in EZLNTFYLST for a specific resource:
NOTIFY PU0001,MSG=YES,DDF=NO,ALERT=TEC,INFORM=NO,EXIT10=TEXIT10 &RESNAME
EXIT11 and EXIT12 (Inform Policy)

EXIT11 and EXIT12 Inform Policy Processing

Purpose
EXIT11 and EXIT12 enable you to alter the inform policy processing.

EXIT11 is called prior to inform policy processing and can be used to dynamically change the inform policy name, group name, or provide a list of policy or group names as needed to insure the appropriate contacts are made.

EXIT12 is called after inform policy processing and before invoking the interface specified in the policy, enabling the inform action to be changed.

Syntax

```
SETUP

EXIT11= ("preprocessing_exit")
EXIT12= ("postprocessing_exit")
```

Parameters

CONNECTIONS
A list of additionally supported connection types. If an interface supports FAX, then FAX must be added to the SETUP CONNECTIONS parameter. It is not necessary to add NUMPAGE, ALPHAPAGE, or EMAIL to this list.

LOG
Set the LOG keyword to YES if you want the INFORM Log enabled. When enabled, selected INFORM actions are logged, and this list can be displayed using the full screen AON function, ILOG. ILOG enables each INFORM action to be deleted, acknowledged, or reissued as needed. Because I/O is required, the default for SETUP LOG is NO. Actions caused by the use of the INFORM command are not logged.

MEMBER
If LOG=YES is specified, then membername is the member that AON INFORM uses to write the log records. The records are written to the first data set name found in the DSILIST data set definition. If no member name is specified, EZLIFLOG is used by default. The write protection key >INFORM or the first line of the member starting in column 1 is written. Therefore, the INFORM log function overwrites only other INFORM logs.
LOGTASK

The INFORM log requires an autotask for the sequencing of all updates. If the log is enabled, the autotask indicated is started, or AUTOINF is started by default.

SPDOM

The domain ID of the network NetView that owns the INFORM service point or application. The current domain is the default.

EXIT11

EXIT11 is invoked prior to checking the in-storage inform policy information.

preprocessing_exit

Specifies the pre-processing exit for the inform policy. For information on the parameters that are passed, see "Usage".

EXIT12

EXIT12 is called prior to invoking the interface specified in the inform policy.

postprocessing_exit

Specifies the post-processing exit for the inform policy. The parameters passed to EXIT12 are the same as those passed to the INTERFACE routine. For more information on the parameters that are passed, see "Usage".

Usage

When EXIT11 is called from EZLENFRM or a notification policy, it receives the policy name, resource name, resource type, domain ID, resource status, and automation status. When EXIT11 is called from EZLECALL or the INFORM command, it receives the policy name and the domain ID. The return codes that can be set by the exit are:

0  Continue normal inform policy processing.
4  Continue normal inform policy processing using changed information. This new information is specified using a task global variable EZLPOLEX which contains the value policy name or group name or list of common delimited policy and group names.
8  Discontinue processing. This enables linkage to other inform technologies without returning to this inform policy flow.

When EXIT12 is called prior to invoking the inform action routine, it receives information related to the inform action. Refer to the EZLENETF sample in the CNMCLST data set for a description and the format of these parameters. The return codes that can be set by the exit are:

0  Call the interface code as specified in the inform policy contact statement.
"  Discontinue processing. Before returning this value you can call other interface routines directly.
EXIT13 Socket Monitoring

Purpose

EXIT13 enables you to prevent the STOP and ACTIVATE commands from being issued against a socket. It also enables you to override the STOP or ACTIVATE commands.

EXIT13 is called for both STOP and ACTIVATE processing.

Syntax

IPPORT

\[
\begin{align*}
\text{IPPORT Socket\_Name} & \quad , \quad \text{SP=stack\_name} \\
\text{PORT=Port\_Num} & \quad , \quad \text{TCPNAME=\_User\_ID} \\
\text{PROTOCOL=Port\_Type} & \quad , \quad \text{YES\_NO} \\
\text{MVS\_TSO\_UNIX\_NETV} & \\
\text{INTVL=Interval} & \quad , \quad \text{DELAY=Mins} \\
\text{ACTIVATE=\_Start\_Command} & \quad , \quad \text{STOP=\_Stop\_Command} \\
\text{EXIT13=\_User\_Exit} & \quad , \quad \text{DESC=\_User\_Text} \\
\text{ACTMON=\_Actmon\_Def} & \quad , \quad \text{FORMAT=PORT} \\
\text{STATUS=\_status1,....,\_statusn} & 
\end{align*}
\]

Usage

You can use Exit13 to prevent the STOP and ACTIVATE commands from being issued against a socket, or to override the STOP or ACTIVATE commands. Exit13 will be called for both STOP and ACTIVATE processing. The input that is available to this exit includes the following task global variables:
FKX_Command
STOP or ACTIVATE command as coded in the socket policy

FKX_Delay
Delay time between STOP and ACTIVATE commands

FKX_Action
Identifies type of action, STOP or ACTIVATE

FKX_CmdType
Identifies the type of command MVS, NETV, TSO, or UNIX

You can choose the following task global variables with this exit:

FKX_Command
Modifies the STOP or ACTIVATE command

FKX_Delay
Modifies the delay time between the STOP and ACTIVATE commands

FKX_CmdType
Modifies the type of command, MVS, NETV, TSO, or UNIX

When zero (0) is the return code either the STOP or ACTIVATE command executes.

When a nonzero return code is received from the user exit, the STOP command will not be issued, a DELAY is not valid, and the user exit will attempt to process the ACTIVATE command. If the nonzero return code is for an ACTIVATE command, ACTIVATE will not be attempted. AON recovery monitoring will still occur.
EXIT14 SNMP MIB Polling

Purpose
EXIT14 is driven during SNMP MIB polling processing. AON/TCP has retrieved the status of a resource interface table (IFTable).

EXIT14 enables you to perform further processing, such as retrieving the IpAddr Table from the resource to correlate the interfaces of the resource with their IP addresses.

EXIT14 can be coded for any of the IP390 resource definitions in the following list. For complete syntax, refer to the Tivoli NetView for z/OS Administration Reference.
- IPHOST
- IPINFC
- IPNAMESERV
- IPROUTER
- IPTN3270

Note: Because EXIT14 is used during SNMP MIB polling, specify FORMAT=SNMP on your resource definitions.

Syntax

IPROUTER

Parameters
AON_Name
The unique name associated with the router that is used by AON/TCP.
Stack_Name
The name of the MVS stack to use for TCP/IP commands. This name is synonymous with the MVS service point.

OPTION=IP390
The only valid entry is IP390.

IPADDR
Consists of dotted decimal IP addresses up to 17 characters including dots. They must be in the 256.256.256.256 format. Alphabetical characters are not valid.

HOSTNAME
Indicates the fully qualified TCP/IP host name, using up to 30 characters including dots. The host_name variable is case-sensitive. The following is an example of a host name:

mrpres.whitehouse.capital.usa

INTERVAL
Defines the monitoring interval in hh:mm format. This is required for proactive monitoring. If the interval is defined for IPSTAT but not specified monitoring will not occur.

ACTMON_Def
Defines active monitoring for groups of resources in the network. For additional information refer to the Tivoli NetView for z/OS Administration Reference.

FORMAT
Specifies one of the following options to be used to determine the resource status:

PING   Pings the resource to check its status.

SNMP   Uses SNMP MIB polling to check the status of the resource.

STATUS
Defines the expected (AON) status of the resource.

MIBVARn
Multiple MIBVAR statements can be defined and are needed only for SNMP thresholding (requires FORMAT=SNMP).

CORRELATE Y|N
Set to Y for trap correlation of IPHOSTs/IPROUTERS/IPTN3270s and their respective interface. The CORRELATE parameter should only be set to Y for resources when SNMP is available. All known interfaces are used to correlate the status. Correlation is only recommended for interfaces installed on critical IPHOSTs or IPROUTERS. N is the default.

EXIT14
Defines additional processing of SNMP interface table. The default value is NONE. Invoked for FORMAT=SNMP.

EXIT15
Defines additional processing of user defined thresholds (MIBVAR). The default value is NONE.

Example
The following example defines ROUTER1 as part of the NYROUTERS policy grouping to be monitored every 15 minutes through SNMP MIB polling:
EXIT14

ACTMON IP390,OPTION=IP390,INTVL=01:,STATUS=NORMAL
ACTMON IPROUTER,OPTION=IP390,INTVL=00:30,STATUS=NORMAL
ACTMON NYROUTERS,OPTION=IP390,INTVL=00:15,
   FORMAT=SNMP

IPROUTER ROUTER1,
   SP=NMPIPL10,
   OPTION=IP390,
   IPADDR=9.67.2.299,
   HOSTNAME=router1.plant.floor.co,
   ACTMON=NYROUTERS,
   EXIT14=MYEXIT14
EXIT15 SNMP MIB Thresholding

Purpose
EXIT15 is driven during SNMP MIB Thresholding processing. AON/TCP has issued SNMP GET requests for each MIBVARx variable for a given resource.

EXIT15 enables you to perform further processing such as determining which MIB variable and value did not meet the threshold specification.

EXIT15 can be coded for any IP390 resource definition in the following list. For additional information refer to the Tivoli NetView for z/OS Administration Reference.

- IPHOST
- IPINFC
- INPNAMESERV
- IPROUTER
- IPTN3270

IPHOST

AON_Name
The unique name associated with the TCP/IP host that is used by AON/TCP.

MVS_Stack_Name
The name of the MVS stack to use for TCP/IP commands. This name is synonymous with the MVS service point.
EXIT15

OPTION=IP390
Only valid entry is IP390.

IPADDR
Consists of dotted decimal IP addresses up to 17 characters including dots. They must be in the 256.256.256.256 format. Alphabetical characters are not valid.

HOSTNAME
Indicates the fully qualified TCP/IP host name, using up to 30 characters including dots. The host_name variable is case-sensitive. The following is an example of a host_name:
mrpres.whitehouse.capital.usa

INTERVAL
Defines the monitoring interval in hh:mm format. This is required for proactive monitoring.

ACTMON_Def
Defines active monitoring for groups of resources in the network. For additional information, refer to the Tivoli NetView for z/OS Administration Reference.

FORMAT
Specifies one of the following options to be used to determine the status of the resource.

PING   Pings the resource to check its status.

SNMP   Uses SNMP MIB polling to check the status of the resource.

STATUS
Defines the expected (AON) status of the resource.

MIBVAR1 – n
Multiple MIBVAR statements can be defined and is needed only for SNMP thresholding (requires FORMAT=SNMP).

CORRELATE Y’N
Set to Y for trap correlation of IPHOSTs/IPROUTERs/IPTN3270s and their respective interface. The CORRELATE parameter should only be set to Y for resources when SNMP is available. All known interfaces are used to correlate the status. Correlation is only recommended for interfaces installed on critical IPHOSTs or IPROUTERs. N is the default.

EXIT14
Defines additional processing of SNMP interface table. The default value is NONE. Invoked for FORMAT=SNMP.

EXIT15
Defines additional processing of user defined thresholds (MIBVAR). The default value is NONE.

Example:
The following example defines host pkoch for proactive monitoring using SNMP polling, looking for a status of NORMAL, THRESH*, or DEGR*:

ACTMON   IP390,OPTION=IP390,INTVL=01:00,STATUS=NORMAL

IPHOST   PKOCH,SP=NMPIPL10,
OPTION=IP390,
HOSTNAME=pkoch.raleigh.tivoli.com,
INTVL=45
FORMAT=SNMP,
STATUS=(NORMAL,THRESH*,DEGR*),
MIBVAR1=(tcpActiveOpens.0,LT,5000),
MIBVAR2=(tcpInErs.0,GT,5),
MIBVAR3=(lplnDiscards.0,EQ,1),
MIBVAR4=(ipRoutingDiscards.0,GE,3),
EXIT15=MYEXIT15
Chapter 11. AON Option Definition Tables

AON option definition tables specify programs that perform resource-dependent functions. The option definition tables also provide literals used by the AON operator interface to create menus containing only selections available in your installation. Each AON function has an option definition table. Some functions, such as AON/SNA, have several options, such as subarea, APPN, SNBU, and X25, that can be enabled independently. Option-specific definitions in the tables are found on the EZLOPT entries. Definitions that are resource-specific are on the EZLRT entries in the option definition tables.

How AON Uses Option Definition Tables

AON uses the EZLERTVE routine to access the option definition tables. For resource-specific definitions, the option definition tables include EZLRT entries. When the EZLERTVE routine is called to look up an EZLRT table value, it first looks for the keyword of the resource type. If the keyword does not exist for that resource type, EZLERTVE obtains the OPTION keyword for the resource type and looks up that keyword. If the keyword is not found there, and the option has an OPTION keyword, EZLERTVE continues looking for an entry in the parent definition of the option.

For example, if the EZLERTVE routine is called to find EZLRT CDRM RESINFO, it searches table EZLRT for resource type CDRM and keyword RESINFO. The EZLERTVE routine looks for RESINFO in the EZLRT CDRM parameter. If it is not found, the EZLERTVE routine obtains the OPTION value of SA from the EZLRT CDRM parameter. The EZLERTVE routine looks for RESINFO on the EZLOPT SA parameter. Again, if it is not found, the EZLERTVE routine gets the OPTION value of SNA from the EZLOPT SA parameter. The EZLERTVE routine looks for RESINFO on the EZLOPT SNA parameter. If it is still not found, the EZLERTVE routine returns N/A to the calling routine.

The first time RESINFO is found on any parameter, the value found there is returned to the calling routine and the EZLERTVE routine exits.

The option definition table uses a tree structure. AON uses the definition found at the lowest level as follows:

EZLOPT SNA → EZLOPT SA → EZLRT CDRM

For more information about EZLERTVE, see “Chapter 8. Coding Common Routines” on page 141.

Displaying or Replacing a Definition

The values in the option definition tables are placed in NetView common global variables for fast access during automation. The common global variable format for resource type values is EZLRT.resource_type.keyword. The common global variable format for option values is EZLOPT.option.keyword. You can display these common global variables by selecting one of the options on the AON Common Global Editor panel shown in Figure 32 on page 244. To access this panel, type CGED and press Enter.
Table 17. Display selection options

<table>
<thead>
<tr>
<th>To display...</th>
<th>...select...</th>
</tr>
</thead>
<tbody>
<tr>
<td>all the resource type values</td>
<td>1</td>
</tr>
<tr>
<td>all the option values</td>
<td>2</td>
</tr>
<tr>
<td>user-specified values</td>
<td>3</td>
</tr>
</tbody>
</table>

3 and type a search criterion. For example, EZL*,.*, IDENTIFY lists all of the IDENTIFY routines defined by option or by resource type, as shown in Figure 33 on page 245.

Figure 32. AON Common Global Editor Panel
To change the value of an EZLOPT entry, use the change function on this panel and type over the current value with the new value. The value is changed immediately. It does not change the option definition table. If you reload your option definition table or recycle NetView, the definition reverts to what is defined in the option definition table. If you need permanent changes, update the definition and replace it, as described in the next section, “Displaying or Replacing an Option Definition Table.”

**Displaying or Replacing an Option Definition Table**

The AON option definition tables are kept in the NetView DSIPARM data set. You can browse or reload one of these tables into the common global variables used by the AON programs by using the AON Loader Tables panel. To display this panel, type the fast path AON 1.8.6 and press Enter. The panel shown in Figure 34 on page 246 is displayed.
The Loader Tables panel lists the tables that are currently loaded. In Figure 34, option definition tables are installed for the AON base and the SNA Automation and TCP/IP Automation features. The panel also contains a blank table-name entry field where you can type a table name not already listed on the panel. You can use the browse option to display the table in DSIPARM. If you select option 2, the table is retrieved from DSIPARM and the common global variables generated by the table are refreshed from the current table. Refreshing the table places the last load of the table into effect immediately.

Guidelines for Option Definition Table Entries

Follow the guidelines described in this topic to create or modify option definition table entries.

Defining Literals

Quotations are not valid unless the literal consists of more than one token. Semicolons (;) are not valid in literals. For example, ENABLE=YES has no semicolon.

Defining Text

Place text inside single quotation marks. Semicolons (;) are not valid. The following is an example:

CMD="RECYCLE" RESNAME'

Defining Lists

Enclose lists in parentheses. A list can contain a space or comma delimiter, but cannot contain a semicolon (;) as a delimiter. The following is an example:

IST1051=(EZL531,FKVEAIDA(resname restype))

In the previous example, the first element in the list is a literal, and the second element is a function call.
Defining Function Calls

Code function calls in the following format:

\[ \text{keyword} = \text{function}\_\text{name}(\text{parm1} \ \text{parm2} \ \text{parm3}) \]

Spaces are not valid between the function name and the left parentheses. Enclose the parameter list after the function name in parentheses. An empty parameter list is indicated by (). Quotations are not valid in the program call. Use spaces as delimiters in the parameters in the list. Semicolons are not valid in function calls. If one of the parameters sent to the function is a literal, enclose the literal in double quotations, as follows:

\[ \text{keyword} = \text{function}\_\text{name}(\text{parm1} \ "\text{lit2}\" \ \text{parm3}) \]

The function must return data in \[ \text{keyword} = \text{variable} \] format or a numeric return code. Use spaces as delimiters in return data. Enclose values containing spaces in double quotations or parentheses.

For example, a RESINFO call for PU resolves to that defined for SNA:

\[ \text{RESINFO} = \text{FKVEAID2}(\text{resname}). \]

If the resource name in the calling program is PU01, the function call resolves to \[ \text{return}\_\text{string} = \text{FKVEAID2(PU01)} \].

In the previous example, the only parameter being sent to function FKVEAID2 is the name of the resource, what the variable \text{resname} contains in the calling program. The calling program must ensure that \text{resname} has a value. The returning data from FKVEAID2 is stored in \text{Return_string}. The \text{Return_string} would contain a data string such as the following:

\[ \text{'RESTYPE=} \text{PU} \ \text{HIGNODE=} \text{LINE01} \ \text{STATUS=} \text{INACT} \ \text{RESHIER=} \text{"NCP01 NCP LINE01 LINE PU01 CTLR"} \]

If the returning data is numeric, it should be a return code.

Defining Command Calls

Command calls in option definition tables are enclosed entirely in single quotations. The input parameters should use spaces as delimiters for variable names and double quotations for literals. For example:

\[ \text{keyword} = "\text{program}\_\text{name} \ \text{var1} \ \text{var2} \ "\text{lit1}" \ \text{var3}" \]

A program should return only numeric return codes to the calling program.

In the next example, the program name is EZLECAUT; \text{resname} is a variable that takes on whatever value \text{resname} has in the calling program:

\[ \text{CHKAUTO} = "\text{EZLECAUT} \ \text{resname} \ \text{PU}" \]

PU is being sent as a literal. If the \text{resname} had the value of PU01, the call is:

\[ \text{EZLECAUT} \ \text{PU01} \ \text{PU} \]

The program returns a return code for analysis by the calling routine. The program can also alter the common global variables or task global variables (TGLOBALs) used by the calling program to influence the processing in the calling program.

Common Global Variable Format

Common global variables are created in the following format:

\[ \text{Table}\_\text{name}.\text{Entry.}\text{Kwx} \]
Error Checking

The table loader stops if it encounters an error. Any common global variables created at that point are defined. AON checks for the following errors:

- Entry = ".
- There are no keywords.
- There are no closing parentheses.
- Keyword values are uneven.
- A semicolon is missing after an entry.
- Value $x$ exceeds the maximum amount of valid data in a common global variable (currently set at 255).
- The option_definition_table is not in the DSIPARM data set.
- The member is empty.

Error Return Codes

Error message 218 is issued for errors found by EZLEALDR after the syntax check. The following return codes are issued to help you determine the problem with the table being loaded:

5       Missing entry ID
6       Missing key values
7       Missing key parameters
8       Uneven keyword in key value
9       Missing ending semicolon
10      Missing comma
11      Keyword value entry > 255

Format for Option Definition Table Entries

This topic illustrates the syntax of an option definition table entry and describes the parameters that can be used in option definition table entries.
table_name entry

Parameters

table_name

The name of the option definition table.

EZLOPT

The option definition table contains the automation definitions for an entire option or suboption. Define only those keywords that apply. The highest level option does not have an OPTION keyword. The OPTION keyword points to the higher level option of the suboption.

EZLRT

The resource type option definition table contains an entry for every valid resource type under any option or suboption. An OPTION keyword must be defined to identify the option that owns this resource type. There are no current provisions for resource types owned by multiple options.

entry

Option name or resource type under an option.

Option names (EZLOPT) are:

SNA This option provides automation definitions and programs for AON/SNA network operation and automation. This option is specifically for VTAM.

SA This suboption of AON/SNA provides automation definitions and programs for subarea SNA networks. Specifically, NCP, LINE, PU, LU, CDRMS, CDRSCs, and APPLs are included.

APPN This suboption of AON/SNA provides automation definitions and programs for Advanced Peer-to-Peer Networking (APPN) SNA networks. This option is specifically for control points (hosts), end nodes, and network nodes.

SNBU This suboption of AON/SNA provides automatic dial backup for failed PUs between two SNA subarea nodes.

X25 This suboption of SNA provides active monitoring of X25 machines defined in NCP and NCP LUDRPOOL availability.

TCPIP This option provides drop-in tables for operating and monitoring of TCP/IP networks using simple network management protocol (SNMP) for network management.

IP390 This suboption of TCPIP provides tables for the operation and monitoring of IP resources using z/OS CommServer IP.

NVAIX

This suboption of TCPIP provides tables for operating and monitoring TCP/IP networks managed by a Tivoli NetView for UNIX service point reporting to this Tivoli NetView for z/OS.

Resource type names (EZLRT) are:
Keyword associated with the entry in the table.

**Keyword names**

**Description**

**ACTMON=’program_call’**

Called by the EZLERECV routine and AON initialization to actively monitor a network resource for availability. Do not use this routine to call EZLEFAIL when the monitored resource is not in an acceptable state. After calling EZLEFAIL, ACTMON should not reschedule itself because EZLEFAIL starts recovery monitoring and EZLERECV restarts this process when the resource is available again.

If the routine is in a normal status, the program should reschedule itself using the NetView AT or AFTER command.

This program should be called as a program because the results of the processed call and the return code is not checked. This process is particularly important to AON/SNA, APPN, and AON/TCP options.

**AUTOVIEW=’program_call’**

When AutoView is started, the presence of this definition adds the option on the AutoView option panel. If the option with this definition is selected by the user, the program is called to provide the user a custom information screen about the resource managed by that option.

**CHKAUTO=’program_call’**

This program is called to discover whether automated recovery,
tracking, and notification should be performed for a resource. This program determines whether the resource is to be automated.

For most resources, the RECOVERY statement of the control file is processed to determine the value of the AUTO parameter and whether the current time is within an applicable NOAUTO window. It can also involve commands and other program calls to determine whether this is an automated resource. EZLECAUT is currently being used as a generic routine to check the RECOVERY entry in the control file. This program is called by EZLEFAIL and EZLERECV. Its call can be skipped by defining SKIP=C on the call to EZLEFAIL/EZLERECV.

Call CHKAUTO as a command because the return code is checked. A return code of zero (0) means automation is in effect. A return code of 1 means that automation is not in effect. If this call is made as a function, the receiving alphanumeric data back from the function is treated as a return code of zero (0).

\textbf{CHKHIGH='program_call'}

This program is called by EZLEFAIL to check that the higher node for the resource is in a state for which recovery on the resource can occur. The higher node can be in terms of hierarchy, connectivity, network management, or priority. For example, if a Line to a PU is inactive, it is impossible to activate the PU because the physical connection is not present; therefore, the call to CHKHIGH should return a 1 (do not continue processing). This is an example of a connectivity higher node.

\textbf{CHKTHR='program_call'}

This program is called by EZLEFAIL to determine whether threshold has been exceeded. The recommended action for this program is to check the control file THRESHOLDS setting for the resource and analyze the status file error log to determine whether threshold setting has been exceeded. This call in EZLEFAIL can be skipped by coding SKIP=(T) on the EZLEFAIL invocation.

Set return codes to:

- \textbf{RC=0} No threshold has been exceeded.
- \textbf{RC=1} Infrequent threshold has been exceeded.
- \textbf{RC=2} Frequent threshold has been exceeded.
- \textbf{RC=3} Critical threshold has been exceeded.

\textbf{CRITACT='prog/func_call'}

This program is called when a critical threshold (RC=3 from the CHKTCHR command call) has been exceeded in EZLEFAIL. Its purpose is to take action when a critical threshold is exceeded. If processing in the program should not continue, a nonzero return code should be returned. If data is returned from a function call, the return code is treated as if it were zero(0), action that should occur when a critical threshold has been exceeded. In the SA option, a critical threshold exception for a PU causes the PU to be deactivated.

\textbf{ENABLE=Y|N}

The enable flag indicates whether an installed function should be enabled. The flag is checked in EZLEFAIL and EZLERECV. Those programs exit if the ENABLE parameter is set to no (ENABLE=N) for the option. This flag is not valid at the EZLRT table level.

\textbf{FREQACT='prog/func_call'}

This program is called when a frequent threshold (RC=3 from the
CHKTHR command call) has been exceeded in EZLEFAIL. Its purpose is to take action when a frequent threshold is exceeded. If processing in the program should not continue, a nonzero return code is returned. If data is returned from a function call, the return code is treated as if it were zero (0). Issue actions that should occur when a frequent threshold has been exceeded

HELPDESK='program_call'
The presence of this definition adds the option into the AON help desk menu panel (EZLK1000). If the option is selected, this program is called to provide a help desk problem determination function for the operator.

IDENTIFY='prog/func_call'
This program is called at the beginning of EZLEFAIL and EZLERECV to determine which option a resource belongs to and what its resource type is. The only parameter that the IDENTIFY program should require is the resource name. The IDENTIFY program should be able to determine whether the resource belongs to this option (where the IDENTIFY program is coded) and what the resource type is. If the resource type is to be returned, the IDENTIFY program should be a function call returning RESTYPE=restype. The return code is assumed to be zero and processing continues. If the return code is not zero, the resource is not managed by this option, and the calling program must try another option or discontinue processing.

INFRACT='prog/func_call'
This program is called when an infrequent threshold (RC=3 from the CHKTHR command call) has been exceeded in EZLEFAIL. Its purpose is to take action when an infrequent threshold has been exceeded. If processing in the program should not continue, a nonzero return code should be returned. If data is returned from a function call, then the return code is treated as if it were zero (0). Issue the action that should occur when an infrequent threshold has been exceeded from this program.

MAINPANELPOS=selection_number_on_AON_panel
This literal is used to format the main operator interface panel. This literal specifies where, in the list of installed options, this selection is displayed. This keyword is used by the 3270 operator interface only and is only valid in the EZLOPT table.

MESSAGING='prog/func_call'
This program is called when operator messaging, logging, and DDF updates are done. It is called from EZLEASLN. You can set special variables for messaging, resource types can be initialized, special processing for updating a particular option, suboption, or resource type can be done.

MSGCLASS=(msgclass_number,...)
MSGCLASS is a 2-digit number used to assign AON notifications to notification operators (as assigned on the NTFYOPS control file entry). Notification operators use this msgclass_number in their CLASS list (on the NTFYOPS control file entry) to receive the message. A message accumulates message classes from the EZLRT MSGCLASS= table entry, EZLOPT MSGCLASS entry for the suboption and option, and the call to search pppMTxx (xx is the first two digits of the message number and ppp is the message prefix, pppxxns). If an entry in this DSIPARM
member has this message ID starting in column one, the MSGCLASS
defined for it is added to the MSGCLASS list.

**OCMDCMD**=**program_name**
Use this command call to issue the 3270 operator interface for an
option or suboption. This keyword is used by the 3270 operator
interface only and is only valid in the EZLOPT table.

**OCMDDESC**="**literal_for_panel_id**"
This literal is displayed on the main panel of the AON operator
interface menu (EZLR0000) to describe the installed options and
suboptions. This keyword is used by the 3270 operator interface only
and is only valid in the EZLOPT table.

**OPERLIST**=(**operglob1** **operglob2** **operglob3** ... **operglobn**)
Use OPERLIST when allocating work to automation operators to
enable multithreading of the automation work load. If the automation
operator is not active when a command is routed to an automation
operator, it is routed to the next automation operator in the list. If a
command is routed to **operglob2**, it is shipped to the operator ID stored
the common global variable, **operglob2**. Operator IDs are defined in the
AUTOOPS entries in the control file. Refer to the [Tivoli NetView for
z/OS Administration Reference](#) for more information. If operator2 (from
operglob2) is inactive, the command is routed to operglob3. The routine
continues until an active automation operator is found to issue the
command or operglobn is reached.

**OPTION**=**immediate_superoption_owning_the_suboption_or_resource_type**
This literal should reflect the immediate owner of this option or
resource type. For a suboption, it reflects the option with which it is
shipped (as specified in its EZLOPT definition). For a resource type,
the literal reflects the option or suboption that is responsible for
managing the resource type that is specified in its EZLOPT definition.

For example, the SNA option ships with the following table definitions
for its options and suboptions:

```
EZLOPT AON,ENABLE=Y,...
EZLOPT SNA,OPTION=AON,ENABLE=Y,...
...  
EZLOPT SA,OPTION=SNA,ENABLE=Y,...
EZLRT NCP,OPTION=SA,...
EZLRT PU,OPTION=SA,...
...
EZLOPT APPN,OPTION=SNA,ENABLE=Y,...
EZLRT CP,OPTION=APPN,...
...
EZLOPT SNBU,OPTION=SNA,ENABLE=Y,...
EZLRT MODEM,OPTION=SNBU,...
...
```

The AON option does not have an OPTION parameter because it is the
highest level in the EZLOPT table for this grouping.

This keyword is required on every entry except the highest option
level.

**RECOVMON**="**program_call**"
This program is called by EZLEFAIL as the last action before notifying
operations of the failure. RECOVMON date and time entries are given
in GMTDATE and GMTTIME parameters. This command call can be skipped by coding SKIP=(R) on the EZLEFAIL call. Recommended actions for this program are:

1. Check the automation flag to ensure automation has not been turned off for this resource since the last execution of this program.
2. Check the status of the resource to ensure it is still not in an acceptable status (it may have recovered since the last execution of this program).
3. Take action to return the resource to an acceptable status.
4. Post an availability message to the logs.
5. If recovery was not successful, notify operators that the resource is still inactive. If recovery was successful, write a message to the log saying that recovery was successful.
6. Reschedule this program to run on a user-defined interval, typically, the control file MONIT intervals, although some options may choose to use a constant interval. Customers can define the interval without modifying the program.

This should be a command call because the return string and return codes are not checked. It is an asynchronous monitoring process.

**REPORTER=variable_name**

REPERTER is a variable that completes the Reported By field used by AON and some AON function messages. The default variable used is DOMAINID. AON/TCP uses SP.

**RESINFO='function_call'**

The program returns all information required by the calling program about a resource. The only parameters required should be the resource name and perhaps the resource type (if known). A function call passes back data. The return string should be in the “varname=varvalue...” format. The calling program assigns the variables when processing the return string.

**RESLIST='program_call'**

The operator interface calls this program to create a select list of resources of a particular type or for an option (depending on where in the tables it is coded). This keyword is used by the 3270 operator interface only and is only valid in the EZLOPT table.

**STSCMD=program_name**

This is the name of the command processor used by an option to update the AON VSAM status file.

**STSPIPE=program_name**

This is the name of the command processor used by an option to update the AON VSAM status file (PIPE version).

**SUMCMD=program_name**

This is the program name called by the operator interface to create a resource information summary panel for a resource type or option. This keyword is used by the 3270 operator interface only and is only valid in the EZLOPT table.

**tblkey_value**

The value on the TBLKEY= keyword specifies optional processing values used by the EZLEFAIL or EZLERECV routine. If you do not specify the TBLKEY parameter for the EZLEFAIL or EZLERECV
routine, no optional processing or notification occurs. The values on
the TBLKEY parameter specify keywords found in the option
definition tables. In the option definition table, the keywords define the
actual processing values used for optional processing. AON saves the
TBLKEY values in the outmsgid and spec_function variables. Message
EZL509I is the default outmsgid for EZLEFAIL. EZL504I is the default
outmsgid for EZLERECV. The value of TBLKEY is in the following
format:

tblkey_value=(outmsgid,spec_function_call)

For example, the EZLEFAIL routine can be called with:

EZLEFAIL OPTION=SA MSGPRMS=(OPID) TBLKEY=IST105I
RESNAME=resname

The EZLEFAIL routine obtains the values specified on the IST105I
keyword in the option definition table, for example:
IST105I=(EZL531,FKVEAIDA(resname restype))

In the previous example, the EZLEFAIL routine issues the EZL531I
message and runs FKVEAIDA as a function sending the current value
of resname (resource name) and restype (resource type) for optional
processing. An optional processing program would perform any
automation or processing unique to the resource or failure. No optional
processing is done and no message is issued if SKIP=(0) is specified on
the EZLEFAIL or EZLERECV call.

The EZLEFAIL routine issues message EZL509I to all logs and to DDF.
The EZLERECV routine issues EZL509I to all logs and to DDF.
Operators do not receive this message. This message is not issued if
SKIP=(A) is specified on the EZLEFAIL or EZLERECV call.

Usage
Sequence numbers in option definition tables can cause unpredictable results.
Part 4. Appendixes
Appendix A. VTAM Messages

This chapter explains the purpose of the VTAM messages that AON/SNA uses, shows the operator response, and any special processing.

General Resource VTAM Messages

**IST093I** resource_name ACTIVE

**IST1132I** resource_name IS ACTIVE, TYPE=resource_type

**Problem Determination:** Indicates when a resource becomes active. AON/SNA posts the resource as available.

**Operator Response:**
- EZL504I : resource_type resource_name IS AVAILABLE
- EZL517I : resource_type resource_name HAS BECOME ACTIVE FROM INTERVENTION BY OPERATOR operator_id

**Explanation:**
If the operator that recovered the resource is an AON/SNA automation operator, suppress the EZL517I operator notification because the IST093I was issued as a result of automation activity.

Clear the threshold settings from the status file in preparation for the next failure.

If recovery is in effect, the status of the resource is not CONCT, and the resource type is a PU, LINE, or LINKSTA, check to see if this is part of a two-line TG with lines of unequal speeds and switch traffic flow to the faster (primary) line and relegate the slower line to a back up function in case the faster line fails. This is defined by the TGSWITCH control file definitions.

**IST105I** resource_name NODE NOW INACTIVE

**IST1133I** resource_name IS NOW INACTIVE, TYPE=resource_type

**Problem Determination:** If VTAM issues IST105I as a result of an operator (OST) command, AON/SNA posts status and stops automated recovery. This enables the operator to work with a resource without AON intervention. The operator is now responsible for recovery of the resource. If VTAM issues the message as unsolicited, AON/SNA initiates standard EZLEFAIL recovery.

**Operator Response:**
- EZL505I : RECOVERY TERMINATED FOR resource_type resource_name DUE TO ACTION BY OPERATOR operator_id
- EZL509I : resource_type resource_name IS UNAVAILABLE
- EZL531I : resource_type resource_name IS INACTIVE DUE TO OPERATOR operator_id INTERVENTION

**Explanation:**
If an operator issues the INACT command, stop recovery and issue the EZL531I.

If a timer already exists for recovery monitoring, purge the timer and start recovery monitoring again.

If the automation status is INRCVY REMIND or REACTV, issue the EZL505I message to all operators.

**IST129I** UNRECOVERABLE OR FORCED ERROR ON NODE resource_name - VARY INACT SCHED

**IST1135I** FORCED VARY INACT SCHEDULED FOR resource_name

**IST1136I** VARY INACT resource_name SCHEDULED - UNRECOVERABLE ERROR

**Problem Determination:** Detects if resource is in a state that AON/SNA cannot recover it from. Stop automation and notify operators that intervention may be required.

**Operator Response:**
- FKV526I : resource_type resource_name IS IN AN INVALID STATE: CURRENT STATUS IS resource_status

**Explanation:** Thresholding, availability, messaging, and recovery are not performed by the EZLEFAIL program.

Stops recovery attempts and messaging if the resource is in an unrecoverable state: not ACT* CON* INA* or IIN*.

**IST383I** DEACTIVATION OF ID = resource_name FAILED - REQUEST: rcmd SENSE : rsense.
Problem Determination:
Detects failed resource deactivations. AON/SNA initiates recovery for the resource with the EZLEFAIL program.

Operator Response: EZL509I : resource_type resource_name IS UNAVAILABLE
FKV527I : DEACTIVATION OF resource_name CANNOT BE COMPLETED BECAUSE resource_name HAS FAILED WITH SENSE: rsense

Explanation: None

Problem Determination:
Detect when AON/SNA cannot recover a resource because its higher node is not active. AON initiates recovery for the higher node and the lower node with the EZLEFAIL program.

Operator Response: FKV529I : ACTIVATION OF minor_resource_name FAILED DUE TO INACTIVE HIGHER NODE - resource_name. ACTIVATION OF resource_name IS ATTEMPTED
EZL509I : resource_type resource_name IS UNAVAILABLE

Explanation: Thresholding is not performed for the resource.

Problem Determination:
Detects a resource failure that VTAM is attempting to recover. AON/SNA enables VTAM MONIT interval to recover the resource. If VTAM is unsuccessful, AON/SNA initiates recovery.

Operator Response: EZL509I : resource_type resource_name IS UNAVAILABLE

Explanation: Thresholding and recovery is performed after the MONIT interval delay.

Problem Determination: Detects successful VTAM recovery of a resource. Posts active resource status and discontinues AON/SNA recovery efforts.

Operator Response: EZL504I : resource_type resource_name IS AVAILABLE

Explanation: Clear the status file THRSHLD setting if it is CRIT.

If RECOVERY flags are on for the resource (call EZLECAUT), the resource status is not CON* or REC*, and the resource type is a PU, LINE, or LINKSTA, check to see if the resource is one of a two line TG with lines of unequal speeds. Manage the line so that the faster line (the primary line) is carrying all the traffic and relegate the slower line to a back up function in case the faster line fails. This is defined by the control file TGSWITCH definitions.

CDRM VTAM Messages

IST727I COMMUNICATION WITH CDRM resource_name LOST - REASON=X'cause code'.

Problem Determination: Detects and notifies operators of a CDRM failure. AON/SNA tries to recover CDRM failures with the EZLEFAIL program.

Operator Response: FKV520I : COMMUNICATION WITH CDRM resource_name LOST DUE TO FORCED INACTIVATE OF THE VR (RC = rc); AUTOMATIC RECOVERY IN PROGRESS
FKV521I : COMMUNICATION WITH CDRM resource_name LOST DUE TO VR INOP (RC = cause_code); AUTOMATIC RECOVERY IN PROGRESS
FKV522I : COMMUNICATION WITH CDRM resource_name LOST DUE TO SSCP - FAILURE (RC = rc); AUTOMATIC RECOVERY IN PROGRESS
FKV525I : COMMUNICATION WITH CDRM resource_name LOST DUE TO SESSION OVERRIDE - ACTIVATE ALREADY IN PROGRESS (RC = cause_code.)
FKV511I : COMMUNICATION WITH CDRM resource_name LOST DUE TO SSCP FAILURE - (RC=rc); AUTOMATIC RECOVERY IN PROGRESS
FKV517I : COMMUNICATION WITH CDRM resource_name LOST DUE TO CLEANUP - THE SSCP IS RESETTING (RC = cause_code)
FKV519I : COMMUNICATION WITH CDRM resource_name LOST DUE TO SSCP CONTENTION (RC = cause_code)
FKV531I : COMMUNICATION WITH CDRM resource_name LOST DUE TO GATEWAY NODE CLEANUP (RC = cause_code)
Explanation: EZL504I is not issued for the CDRM by the EZLEFAIL program.

IST742I ACTIVATION OF CDRM resource_name FAILED GWN PATH NOT AVAILABLE

Problem Determination: Detects and notifies operators of a CDRM failure. AON/SNA initiates CDRM recovery with the EZLEFAIL program.

Operator Response: FKV541I : ACTIVATION FAILED FOR CDRM resource_name; GATEWAY PATH NOT AVAILABLE

Explanation: Thresholding and recovery not run by the EZLEFAIL program.

Application Messages

IST400I TERMINATION IN PROGRESS FOR APPLID resource_name

Problem Determination: Detects and notifies operators that the application stopped.

Operator Response: FKV552I : APPLICATION resource_name HAS BEEN TERMINATED BY VTAM

Explanation: Availability messaging thresholding and recovery are not run by the EZLEFAIL program.

IST804I VTAM CLOSE IN PROGRESS FOR applname OPENED BY jobname

Problem Determination: Detects a VTAM ACB close has started. AON/SNA monitors the process to be sure that the close is successful. IST805I is issued. This prevents hung ACBs.

Operator Response: EZL550I : APPLICATION appl WAS NOT CLOSED CORRECTLY BY operator_id

Explanation: EZLEFAIL processing for availability messages, thresholding, messaging, and recovery is not done.

Check the control file recovery flag for JOBNAME. If recovery is off for it then exit.

Host VTAM Messages

IST348I UNABLE TO PROCESS DISCONNECTION FOR PU = resource_name DUE TO LACK OF STORAGE

Problem Determination: Notify operators of storage shortage problems during disconnection of a PU.

Operator Response:
EZL509I : resource_type resource_name IS UNAVAILABLE
FKV518I : resource_type resource_name DISCONNECTION FAILED DUE TO LACK OF STORAGE - STATUS IS resource_status

Explanation: ‘D NET,BFRUSE’

IST561I STORAGE UNAVAILABLE: pool BUFFER POOL

Problem Determination: Detects and notifies operators of storage shortage problems.

Operator Response:
FKV514I : STORAGE UNAVAILABLE FOR resource_name BUFFER POOL

Explanation: ‘D NET,BFRUSE’

The EZLEFAIL program performs only optional processing and messaging.
**Problem Determination:** Detects and notifies operators of storage shortage problems in CSA.

**Operator Response:** FKV515I : STORAGE UNAVAILABLE - resource_name HAS BEEN REACHED

**Explanation:** 'D NET,BFRUSE'
The EZLEFAIL program performs only optional processing and messaging.

**Problem Determination:** Detects and notifies operators of storage shortage problems in CSA.

**Operator Response:** FKV516I : STORAGE UNAVAILABLE FOR SUBPOOL resource_name

**Explanation:** 'D NET,BFRUSE'
The EZLEFAIL program performs only optional processing and messaging.

**Problem Determination:** Detects disconnection failure. AON/SNA initiates recovery with the EZLEFAIL program.

**Operator Response:** EZL509I : resource_type resource_name IS UNAVAILABLE

**Explanation:** The EZLEFAIL program only runs messaging.

**Problem Determination:** Detects NCP option to dump WTOR. AON responds to the WTOR according to the NCPRECOV definitions. EXIT01 is used. A timer is set to ensure that the dump does not take longer than the time specified in the DUMPTIME parameter. The EZLEFAIL program is not used.

**Operator Response:** EZL509I : resource_type resource_name IS UNAVAILABLE

**Explanation:** The EZLEFAIL program only runs messaging.

**Problem Determination:** Finishes NCP recovery by indicating that the load of the NCP is complete. AON/SNA issues EXIT04. The timer for LOADTIME is purged because the load of the NCP is now complete. The EZLEFAIL program is not used.

**Operator Response:** FKV544I : RELOAD WAS SUCCESSFUL FOR resource_name AND IS AVAILABLE

**Explanation:** The EZLEFAIL program only runs messaging.

**Problem Determination:** Detects and notifies operators of a storage shortage problem.

**Operator Response:** FKV513I : STORAGE UNAVAILABLE FOR ADJSSCP TABLE FOR resource_name

**Explanation:** 'D NET,BFRUSE'
The EZLEFAIL program performs only optional processing and messaging.

**Problem Determination:** Detects disconnection failure. AON/SNA initiates recovery with the EZLEFAIL program.

**Operator Response:** FKV524I : SESSION LOST BETWEEN resource_name AND sscp_name IN SA sa
FKV539I : SESSION LOST BETWEEN resource_name AND reply IN SA sa DUE TO A FORCED DEACTIVATION OF THE SSCP-PU SESSION
FKV543I : SESSION LOST BETWEEN resource_name AND resource_name2 DUE TO DEACTIVATION OF THE VIRTUAL ROUTE
FKV545I : SESSION LOST BETWEEN resource_name AND resource_name2 IN SA sa DUE TO AN SSCP FAILURE

**Explanation:** The EZLEFAIL program only runs messaging.

**Problem Determination:** Detects disconnection failure. AON/SNA initiates recovery with the EZLEFAIL program.

**Operator Response:** FKV558I : DISCONNECT OF resource_name FAILED DUE TO I/O ERROR OR INSUFFICIENT STORAGE

**Explanation:** 'D NET,BFRUSE'

**Problem Determination:** Detects disconnection failure. AON/SNA initiates recovery with the EZLEFAIL program.

**Operator Response:** FKV538I : REPLY OF reply WAS ISSUED BY AUTOMATION FOR ncpname FROM host: CRITICAL RELOAD FROM NON-RECOVERY HOST
FKV535I : REPLY OF reply WAS ISSUED BY AUTOMATION FOR ncpname FROM host: NON-CRITICAL RELOAD FROM NON-RECOVERY HOST
FKV528I : DISCONNECT OF resource_name FAILED DUE TO I/O ERROR OR INSUFFICIENT STORAGE

**Explanation:** The EZLEFAIL program only runs messaging.

**Problem Determination:** Detects NCP option to dump WTOR. AON responds to the WTOR according to the NCPRECOV definitions. EXIT01 is used. A timer is set to ensure that the dump does not take longer than the time specified in the DUMPTIME parameter. The EZLEFAIL program is not used.

**Operator Response:** EZL509I : resource_type resource_name IS UNAVAILABLE

**Explanation:** The EZLEFAIL program only runs messaging.

**Problem Determination:** Detects NCP option to dump WTOR. AON responds to the WTOR according to the NCPRECOV definitions. EXIT01 is used. A timer is set to ensure that the dump does not take longer than the time specified in the DUMPTIME parameter. The EZLEFAIL program is not used.

**Operator Response:** EZL509I : resource_type resource_name IS UNAVAILABLE

**Explanation:** The EZLEFAIL program only runs messaging.

**Problem Determination:** Detects NCP option to dump WTOR. AON responds to the WTOR according to the NCPRECOV definitions. EXIT01 is used. A timer is set to ensure that the dump does not take longer than the time specified in the DUMPTIME parameter. The EZLEFAIL program is not used.

**Operator Response:** EZL509I : resource_type resource_name IS UNAVAILABLE

**Explanation:** The EZLEFAIL program only runs messaging.

**Problem Determination:** Detects NCP option to dump WTOR. AON responds to the WTOR according to the NCPRECOV definitions. EXIT01 is used. A timer is set to ensure that the dump does not take longer than the time specified in the DUMPTIME parameter. The EZLEFAIL program is not used.

**Operator Response:** EZL509I : resource_type resource_name IS UNAVAILABLE

**Explanation:** The EZLEFAIL program only runs messaging.
IST272A  

**Problem Determination:** Responds to the bypass initial load WTOR for the NCP. Replies U. AON/SNA uses the EZLEFAIL program to run.

**Operator Response:** EZL509I : resource_type
resource_name IS UNAVAILABLE

FKV530I : BYPASS THE INITIAL TEST ROUTINE FOR resource_name - REPLY "U" TO BYPASS WAS ISSUED

FKV551I : REPLY FOR BYPASS INITIAL TEST FOR NCP resource_name NOT ISSUED; AUTOMATION FOR resource_name IS OFF: REPLY 'U' TO BYPASS OR CANCEL FOR REPLY ID reply

**Explanation:** AON/SNA runs only special processing in the EZLEFAIL program to respond to the outstanding reply.

IST278A  

**Problem Determination:** Responds to a bad reply to IST272A. Replies U.

**Operator Response:** EZL509I : resource_type
resource_name IS UNAVAILABLE

FKV530I : BYPASS THE INITIAL TEST ROUTINE FOR resource_name - REPLY "U" TO BYPASS WAS ISSUED

FKV551I : REPLY FOR BYPASS INITIAL TEST FOR NCP resource_name NOT ISSUED; AUTOMATION FOR resource_name IS OFF: REPLY 'U' TO BYPASS OR CANCEL FOR REPLY ID reply

**Explanation:** AON/SNA runs only special processing in the EZLEFAIL program to respond to the outstanding reply.

IST284A  

**Problem Determination:** Responds to the option to reload NCP WTOR according to the NCPRECOV definitions. AON/SNA uses EXIT03 and sets a timer to ensure that the time taken to reload the NCP does not exceed LOADTIME. The EZLEFAIL program is not used.

**Operator Response:** FZL227E : idnet COULD NOT FIND EXPECTED CONFIGURATION DATA FROM NCPRECOV COMMAND : errmsg

FKV551I : REPLY FOR BYPASS INITIAL TEST FOR NCP resource_name NOT ISSUED; AUTOMATION IS OFF: REPLY 'U' TO BYPASS OR CANCEL FOR REPLYID replyid

**Explanation:** Thresholding and recovery are not run in the EZLEFAIL program.

IST361A  

**Problem Determination:** Notify operators when the NCP load module does not match the currently loaded module. AON/SNA replies NO if the NCP is to be automatically recovered.

**Operator Response:** EZL509I resource_type
resource_name IS UNAVAILABLE

FKV510I LOAD OF NCP resource_name STOPPED BECAUSE LOAD MODULE DOES NOT MATCH NEW
NCP; AUTOMATION IS OFF, REPLY 'NO' TO CANCEL OR 'YES' TO RELOAD FOR REPLYID replyid

FKV512I: LOAD OF NCP resource_name CANCELLED - LOAD MODULE DOES NOT MATCH NEW NCP

Explanation: The EZLEFAIL program runs the only optional processing.

FV580I ERROR FOR ID = resource_name FAILED - REQUEST: rcmd SENSE: rsense

Problems Determination: If rcmd is REQDUMP, AON/SNA purges the dump time timer. If rcmd is REQLOAD, AON/SNA purges the load time timer. Notification operators are notified that the LOAD/DUMP has failed. The EZLEFAIL program is not used.

Operator Response: FKV501I DUMP OF NCP FAILED - action reason
FKV502I LOAD OF NCP FAILED - action reason

Explanation: None.

IST584I LINK STATION resource_name1 has CONTACTED resource_name2 SA subarea

Problem Determination: Detects a link station connection to the NCP from AON/SNA. AON/SNA posts link station and NCP as available if the NCP is an automated NCP with a NCPRECOV statement. DUMP and LOAD timers are purged. The EZLEFAIL program is not used.

Operator Response: EZL504I resource_type resource_name IS AVAILABLE
FKV548I resource_name HAS CONTACTED resource_type resource_name - resource_name IS AVAILABLE

IST530I ru PENDING FROM resource TO resource FOR resource_name

IST1278I ru PENDING FROM netid to netid FOR resource_name

Problem Determination: Detects an NCP unavailability from a channel attached non-recovery host. If the NCP status indicates that it is being recovered by another host, AON/SNA enacts/acts the NCP to recover connection to this host. Otherwise, the operator AON/SNA notifies the operator, who manually recovers the NCP. The EZLEFAIL program is not used.

Operator Response: EZL504I resource_type resource_name IS AVAILABLE

IST529I REASON = reason_text

Problem Determination: Detects when the load of an NCP failed. The EZLEFAIL program is called from another program which parses and passes the second IST523I message of the MLWTO.

Operator Response: FKV560I LOAD OF ncp_resource_name FAILED - RECEIVE .reason_text

Explanation: Thresholding and recovery are not run by the EZLEFAIL program.
Appendix B. Sample AON/SNA SNBU Modem Configurations

This section illustrates sample modem configurations to help you code appropriate entries in the control file and select appropriate modem options. The examples provided here are not the only modem options available. Check these modems with the modems listed in the manual for that particular model of IBM modem. Table 18 identifies resources used in all of the following sample configurations.

Note: Ensure that the coupler is installed in the modem and that the phone cable is plugged into the modem.

Table 18. Resource Configuration Key

<table>
<thead>
<tr>
<th>Resource</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>TA1L2005</td>
<td>Port on 37xx defined as tailed circuit</td>
</tr>
<tr>
<td>TA1L2006</td>
<td>Port on 37xx defined as multipoint link</td>
</tr>
<tr>
<td>TA1L2007</td>
<td>Port on 37xx defined as point-to-point link</td>
</tr>
<tr>
<td>7861-047</td>
<td>Local modem</td>
</tr>
<tr>
<td>5866-2</td>
<td>Local modem</td>
</tr>
<tr>
<td>Phone line</td>
<td>Multipoint nonswitched line</td>
</tr>
<tr>
<td>Phone line</td>
<td>Point-to-point nonswitched line</td>
</tr>
<tr>
<td>Tailing cable</td>
<td>Attached to modem R3</td>
</tr>
<tr>
<td>7861-047</td>
<td>Remote modem</td>
</tr>
<tr>
<td>5866-2</td>
<td>Remote modem</td>
</tr>
<tr>
<td>5866-2</td>
<td>Remote modem</td>
</tr>
<tr>
<td>TA1P26A</td>
<td>IBM 3x74 control unit (ADDR=C1)</td>
</tr>
<tr>
<td>TA1P27A</td>
<td>IBM 3x74 control unit (ADDR=C2)</td>
</tr>
<tr>
<td>6431</td>
<td>Public telephone line for SNBU (remote)</td>
</tr>
<tr>
<td>6432</td>
<td>Public telephone line for SNBU (remote)</td>
</tr>
</tbody>
</table>

Point-to-Point Sample Configuration—Same Modem

The following diagram shows a point-to-point sample configuration using the same modem.

Figure 35. Point-to-Point Sample Configuration—Same Modem
Table 19. Modem Configuration for Point-to-Point—Same Modem

<table>
<thead>
<tr>
<th>Function</th>
<th>Local Modem</th>
<th>Remote Modem</th>
</tr>
</thead>
<tbody>
<tr>
<td>Address</td>
<td>01</td>
<td>C2</td>
</tr>
<tr>
<td>Speed</td>
<td>9600</td>
<td>9600</td>
</tr>
<tr>
<td>LPDA2</td>
<td>ENABLED</td>
<td>ENABLED</td>
</tr>
<tr>
<td>Feature</td>
<td>FIFO</td>
<td>FIFO</td>
</tr>
<tr>
<td>Control</td>
<td>PTP P</td>
<td>PTP P</td>
</tr>
<tr>
<td>XMIT clock</td>
<td>INT</td>
<td>RCV</td>
</tr>
</tbody>
</table>

Control file entry:
SNBU TA1P27A,AUTOSW=Y,AUTOBK=Y,PH=(6431,6432),RECONN=Y

Point-to-Point—Modem Pool

The following diagram shows a point-to-point modem pool.

![Diagram of a point-to-point modem pool](image)

Table 20. Modem Configuration for Point-to-Point—Modem Pool

<table>
<thead>
<tr>
<th>Function</th>
<th>Local modem</th>
<th>Pooled modem</th>
<th>Remote modem</th>
</tr>
</thead>
<tbody>
<tr>
<td>Address</td>
<td>01</td>
<td>01</td>
<td>C2</td>
</tr>
<tr>
<td>Speed</td>
<td>9600</td>
<td>9600</td>
<td>9600</td>
</tr>
<tr>
<td>LPDA2</td>
<td>ENABLED</td>
<td>ENABLED</td>
<td>ENABLED</td>
</tr>
<tr>
<td>Feature</td>
<td>FIFO</td>
<td>—</td>
<td>FIFO</td>
</tr>
<tr>
<td>Control</td>
<td>PTP P</td>
<td>PTP P</td>
<td>PTP P</td>
</tr>
<tr>
<td>XMIT clock</td>
<td>INT</td>
<td>INT</td>
<td>RCV</td>
</tr>
</tbody>
</table>

Control file entry:
SNBU TA1P27A,AUTOSW=Y,AUTOBK=Y,PH=(6431,6432),POOL=PU2,RECONN=Y
SNBUPOOL PU2
PU2 TA1L2006

Note: With this entry, AON/SNA SNBU first attempts to use the same SNBU modem first. The modem pool is used only if the same AON/SNA SNBU modem fails. Specify APO=Y (Alternate Port Only) on the AON/SNA SNBU control file entry to force the use of the modem pool only.
Multipoint Sample Configuration—Modem Pool

The following diagram shows a multipoint sample configuration modem pool.

![Diagram of Multipoint Sample Configuration—Modem Pool]

Table 21. Modem Configuration for Multipoint—Modem Pool

<table>
<thead>
<tr>
<th>Function</th>
<th>Local modem</th>
<th>Pooled modem</th>
<th>Remote modem 1</th>
<th>Remote modem 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Address</td>
<td>01</td>
<td>01</td>
<td>C2</td>
<td>C1</td>
</tr>
<tr>
<td>Speed</td>
<td>9600</td>
<td>9600</td>
<td>9600</td>
<td>9600</td>
</tr>
<tr>
<td>LPDA2</td>
<td>ENABLED</td>
<td>ENABLED</td>
<td>ENABLED</td>
<td>ENABLED</td>
</tr>
<tr>
<td>Feature</td>
<td>FIFO</td>
<td>—</td>
<td>FIFO</td>
<td>—</td>
</tr>
<tr>
<td>Control</td>
<td>MTP C</td>
<td>MTP C</td>
<td>MTP T</td>
<td>MTP T</td>
</tr>
<tr>
<td>XMIT clock</td>
<td>INT</td>
<td>–</td>
<td>RCV</td>
<td>RCV</td>
</tr>
</tbody>
</table>

Control file entry:

```
SNBU TA1P27A,AUTOSW=Y,AUTOBK=Y,PH=(6431,6432),POOL=PU2,APO=Y,RECONN=Y
SNBUPOOL PU2
PU2 TAIL2007
```

Note: APO=Y (alternate port only) indicates that only the pool modems are selected for AON/SNA SNBU.

Fanout Sample Configuration—Same Modem

The following diagram shows a fanout sample configuration on the same modem.
Table 22. Modem Configuration for Fanout—Same Modem

<table>
<thead>
<tr>
<th>Function</th>
<th>Local modem</th>
<th>Remote modem</th>
</tr>
</thead>
<tbody>
<tr>
<td>Address</td>
<td>01</td>
<td>C2</td>
</tr>
<tr>
<td>Speed</td>
<td>9600</td>
<td>9600</td>
</tr>
<tr>
<td>LPDA2</td>
<td>ENABLED</td>
<td>ENABLED</td>
</tr>
<tr>
<td>Feature</td>
<td>FIFO</td>
<td>FIFO</td>
</tr>
<tr>
<td>Control</td>
<td>MTP C</td>
<td>MTP T</td>
</tr>
<tr>
<td>XMIT clock</td>
<td>INT</td>
<td>RCV</td>
</tr>
</tbody>
</table>

Control file entry:

SNBU TA1P27A,AUTOSW=Y,AUTOBK=Y,PH=(6431,6432),FANOUTS=TA1P26A,RECONN=Y

Fanout Sample Configuration—Modem Pool

The following diagram shows a fanout sample configuration modem pool.
Table 23. Modem Configuration for Fanout—Modem Pool

<table>
<thead>
<tr>
<th>Function</th>
<th>Local modem</th>
<th>Pooled modem</th>
<th>Remote modem</th>
</tr>
</thead>
<tbody>
<tr>
<td>Address</td>
<td>01</td>
<td>01</td>
<td>C2</td>
</tr>
<tr>
<td>Speed</td>
<td>9600</td>
<td>9600</td>
<td>9600</td>
</tr>
<tr>
<td>LPDA2</td>
<td>ENABLED</td>
<td>ENABLED</td>
<td>ENABLED</td>
</tr>
<tr>
<td>Feature</td>
<td>FIFO</td>
<td>—</td>
<td>FIFO</td>
</tr>
<tr>
<td>Control</td>
<td>MTP C</td>
<td>MTP C</td>
<td>MTP T</td>
</tr>
<tr>
<td>XMIT clock</td>
<td>INT</td>
<td>INT</td>
<td>RCV</td>
</tr>
</tbody>
</table>

Control file entry:

```
SNBU TA1P27A,AUTOSW=Y,AUTOBK=Y,PH=(6431,6432),POOL=PU2,APO=Y,
FANOUTS=TA1P26A,RECONN=Y
SNBUPOOL PU2
PU2 TA1L2007
```

Multiport (DMPX) Sample Configuration—Same Modem

The following diagram shows a multiport (DMPX) sample configuration on the same modem.

![Multiport (DMPX) Sample Configuration—Same Modem](image)

Figure 40. Multiport (DMPX) Sample Configuration—Same Modem

An alternate port AON/SNA SNBU is not supported for DMPX.

Table 24. Modem Configuration for Multiport (DMPX)—Same Modem

<table>
<thead>
<tr>
<th>Function</th>
<th>Local modem</th>
<th>Remote modem</th>
</tr>
</thead>
<tbody>
<tr>
<td>Address</td>
<td>01</td>
<td>C2</td>
</tr>
<tr>
<td>Speed</td>
<td>19200</td>
<td>19200</td>
</tr>
<tr>
<td>LPDA2</td>
<td>ENABLED</td>
<td>ENABLED</td>
</tr>
<tr>
<td>Feature</td>
<td>DMPX</td>
<td>DMPX</td>
</tr>
<tr>
<td>Control</td>
<td>PTP P</td>
<td>PTP S</td>
</tr>
<tr>
<td>XMIT clock</td>
<td>INT</td>
<td>RCV</td>
</tr>
<tr>
<td>DMPX Options A FULL SPD</td>
<td>9600</td>
<td>9600</td>
</tr>
</tbody>
</table>
Control file entry:
SNBU TA1P27A,AUTOSW=Y,AUTOBK=Y,PH=(6431,6432),FANOUTS=TA1P26A,RECONN=Y

**Tailing—Same Modem (LSL1)**

The following diagram shows a detailed modem configuration with AON/SNA SNBU enabled for line segment for link segment level 1:

![Diagram of Tailing—Same Modem (LSL1)](image)

**Figure 41. Tailing—Same Modem (LSL1)**

**Table 25. Modem Configuration for Tailing—Same Modem (LSL1)**

<table>
<thead>
<tr>
<th>Function</th>
<th>Local modem—Level 1</th>
<th>Remote modem—Level 1</th>
<th>Remote modem—Level 2</th>
<th>Remote modem—Level 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Address</td>
<td>01</td>
<td>C2</td>
<td>02</td>
<td>C2</td>
</tr>
<tr>
<td>Speed</td>
<td>9600</td>
<td>9600</td>
<td>9600</td>
<td>9600</td>
</tr>
<tr>
<td>Speed CTL</td>
<td>MODEM</td>
<td>MODEM</td>
<td>DTE</td>
<td>MODEM</td>
</tr>
<tr>
<td>LPDA2</td>
<td>ENABLED</td>
<td>ENABLED</td>
<td>ENABLED</td>
<td>ENABLED</td>
</tr>
<tr>
<td>Feature</td>
<td>FIFO</td>
<td>FIFO</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>Control</td>
<td>PTP P</td>
<td>PTP S</td>
<td>PTP P</td>
<td>PTP S</td>
</tr>
<tr>
<td>XMIT clock</td>
<td>INT</td>
<td>RCV</td>
<td>EXT</td>
<td>RCV</td>
</tr>
</tbody>
</table>

Control file entry:
SNBU TA1P27A,AUTOSW=Y,AUTOBK=Y,PH=(6431,6432),RECONN=Y,LEVEL=1
DDF (continued)
initial panel 43
initialization statement 40
initialize 27, 30
input file 27
MARK function 73
operator session, enter 27
overview 3, 15
panel member, load 27
panel parameters 27
performance 95
query 27
resource status 16
samples 71
screen size 30
specific implementation 25
starting 28
statement definitions 37
status information, updating 97
status panel, definition 57
status priority 27
stopping 28
system defaults 27
tree member, load 27
tree structure, define 27
DDF command 27, 98
DDF implementation
AON/SNA, adding 34
AON/TCP, adding 34
control file 33
overview 15
DDFADD command 27, 99
DDFCLEAR command 102
DDFDEL command 27, 104
DDFGROUP control file entry 91
DDFPANEL 27
DDFQRY command 27, 106
DDFTREE command 27, 28, 109
debug record 130
definition, displaying or replacing 243
definition table, retrieving 183
delete DDF elements 102
detail status 42
detail status display 19, 22
disability information xi
dynamic display facility 15
E
E-mail contact x
ENDPANEL parameter 68
ENVIRON SETUP 9
error limit, DDF panel 47
error record 130
EXIT 144
EXIT01 216
EXIT02 216
EXIT03 216
EXIT04 216
EXIT05 218
EXIT06 220
EXIT07 223
EXIT08 225
EXIT09 227
EXIT10 230
EXIT11 232
EXIT12 232
EXIT13 234
EXIT14 236
EXIT15 239
EZALOG command 131
EZLCFG command 124
EZLCFG01 15
customizing 75, 78
loading 10
EZLEIUFW 146
EZLEACKT 147
EZLEADCL 102
EZLEAGEN 148
EZLEAGRN 149
EZLEASN 118, 151
EZLEATDF 145
EZLEATHR 119, 156
EZLECALL 160
EZLECATV 118, 161
EZLECAUT 118, 163
EZLEFAIL 164
EZLEMCOL 170
EZLEMFG 171
EZLENFRM 172
EZLENTFY 173
EZLERAIP 174
EZLERCMD 176
EZLERCV 177
EZLERGWY 179
EZLEROUT 181
EZLERTVE 183
EZLESRMD 185
EZLESTOP 186
EZLESTR 187
EZLEVACT 118, 189
EZLEVINA 190
EZLEMOV 191
EZLEXT07 user exit 224
EZLINIT 27
CHAIN parameter 42
customizing 75, 78
dcolor parameter 54
DPFDESC1 parameter 52
DPFDESC2 parameter 53
DPFKn parameter 50
DSITBL01 7
DSPSTS command 6
DVIPOPER 7
dynamic display facility 15
E
E-mail contact x
ENDPANEL parameter 68
ENVIRON SETUP 9
error limit, DDF panel 47
error record 130
EXIT 144
EXIT01 216
EXIT02 216
EXIT03 216
EXIT04 216
EXIT05 218
EXIT06 220
EXIT07 223
EXIT08 225
EXIT09 227
EXIT10 230
EXIT11 232
EXIT12 232
EXIT13 234
EXIT14 236
EXIT15 239
EZALOG command 131
EZLCFG command 124
EZLCFG01 15
customizing 75, 78
loading 10
EZLEIUFW 146
EZLEACKT 147
EZLEADCL 102
EZLEAGEN 148
EZLEAGRN 149
EZLEASN 118, 151
EZLEATDF 145
EZLEATHR 119, 156
EZLECALL 160
EZLECATV 118, 161
EZLECAUT 118, 163
EZLEFAIL 164
EZLEMCOL 170
EZLEMFG 171
EZLENFRM 172
EZLENTFY 173
EZLERAIP 174
EZLERCMD 176
EZLERCV 177
EZLERGWY 179
EZLEROUT 181
EZLERTVE 183
EZLESRMD 185
EZLESTOP 186
EZLESTR 187
EZLEVACT 118, 189
EZLEVINA 190
EZLEMOV 191
EZLEXT07 user exit 224
EZLINIT 27
CHAIN parameter 42
customizing 75, 78
dcolor parameter 54
DPFDESC1 parameter 52
DPFDESC2 parameter 53
DPFKn parameter 50
EMPTYCOLOR parameter 55
INITSCRN parameter 43
MAXOPS parameter 44
PFKn parameter 48
PRIORITY parameter 57
PRITBLSZ parameter 56
PROPDOWN parameter 46
PROPUP parameter 45
SCREENSZ parameter 41
statements, defining 40
TEMPERR parameter 47
EZLLOG command 130
EZLPNL1 27
customizing 79
EZLPNL5 27, 57
%INCLUDE 69
customizing 75, 78, 84
ENDPANEL parameter 68
PANEL parameter 58
panel statements, defining 57
PFKn parameter 66
STATUSFIELD parameter 60
STATUSTEXT parameter 63
TEXTFIELD parameter 64
TEXTTEXT parameter 65
EZLPNLS 58
EZLSMSU 192
EZLSTS 133
EZLSTS record 138
EZLTDDF 27
EZLTLOG command 131
EZLTRACE 194
EZLTREE 18, 27, 33, 38
customizing 74, 82, 88
updating 78
F
feedback about publications x
field, defining panels 31
file manager, interfacing 119
FKVEXT05 user exit 218
FKVEXT06 user exit 221
FKVPNLS 35
FKVSNBU command 137
FKXECNV 196
FKXPNL5 35
focal point
AON 207
focal point environment 4, 81
framework, command list 116
function key 48
G
GATOPER 7
H
help desk 10
I
implementation
data definition file (DDF) 23
generic 23
specific 25
INFOPER 7
information, accessibility xi
information, disability xi
information record 130
initialization 117
chain detail records, linking 42
color, status descriptor 54, 55
detail function key definition 40
operators, maximum 44

initialization *(continued)*
priority, color ranges 56, 57
screen, initial 43
screen size 41
status word text 45, 46
initialization statement 40
INITSCRN parameter 43
input file
EZLIMIT 27
EZLPNLS 27
EZLTREE 27
IP Trace
programmatic interface 199
IPHOST 239
IPROUTER 236

K
keyboard, shortcut keys xi

L
line status 137
log, automation 6
log file
interface 116

M
manuals
feedback ix
online ix
ordering ix
MARK function 73, 86, 110
MAXOPS parameter 44
menu panel definition 30
message, operator 225
messages
automation table 7
VTAM 259
MessageView 79
modem configuration 265
MSGOPER 7
MSU 7
MSU, sending 192
MVS console 9

N
NCP, recovery 216
NETOPER 7
NETSTAT CHKAUTO parameter 22
NetView domains, defining multiple 22
NetView log 6
network, defining to DDF 24
NMC operator 4
NNT, routing logon information 181
notification operator 8
notification policy 6
NV6KOP 7

O
OIV 6
OIVOPER 7
online publications x
operator
assigning a problem 110
interface 4
maximum number 44
removing DDF assignment 111
operator intervention view (OIV) 174
operator session, DDF 27
option definition table
displaying or replacing 245
entry format 248
entry guidelines 246
overview 243
specifying 243
ordering publications x
organization, book ix

P
panel
common global editor 243
defining 22
definition 29
detail panel, function key
definition 50, 52, 53
detail status display 19, 22
error limit 47
field, defining 31
format 27
formatting messages 171
generic display 94
hierarchy 38
inform policy checking 172
initial panel 43
loader tables 245
loading 28
member, load 27
multiple 73
network status 72
operator MARK 86
parameters 27
screen size 30
setting message color 170
single 75
status 38
status panel
%INCLUDE 69
definition 57
end 68
field, constant 64
function key definition 66
start 58
status component 60
status field text 63
PANEL parameter 58
performance, DDF 95
PF key 48
detail display 50
status panel 66
PRIORITY entity 56, 57
problem, assigning 110
PROPDOWN parameter 46
PROPUP parameter 45
publications
feedback ix
online ix

Q
query 27

R
RECOVERY flag 128
recovery process 5
reminders, resource state 185
resource
activating 189, 227
automation status 223
color 19
deactivating 190, 227
getting information 218
grouping 88, 89
moving 191
priority 19
problem 22
recover 177
recovery 5
recovery example 119
routines 141

S
SCREENSZ parameter 41
sessions, cross-domain
starting 187
stopping 186
shortcut keys, keyboard xi
size, screens 41
SNBU automation status file 137
SNBU modem configuration 265
SNMP RFC conversion 196
socket monitoring 234, 236, 239
STARTEZL DDF command 28
status
color definition 38
field 32
information 27
network 76
panel 98
priority 27
word text 45, 46
status, network 71
status descriptor 79
adding 99
deleting 104
querying 107
status file
description 6
display record 136
interface 116
purge records 9
SNBU records 137
updating 151
updating records 133
status word text 46
STATUSFIELD parameter 60, 63
STOPEZL DDF command 28
system defaults 27

Index 273
| T | task  
  EZLTDDF 27  
  TCP/IP trace  
  programmatic interface 199  
  TCPOPER 8  
  TEMPERR parameter 47  
  TEXTFIELD parameter 64, 65  
  TGLOBAL  
  setting 119, 214  
  threshold 8  
  analysis 220  
  checking 156  
  time, calculating 145  
  timer delay 119  
  Tivoli Customer Support  
  trace, entry and exit 194  
  TRAOPER 8  
  tree structure  
  %INCLUDE 27  
  definition 38  
  example 18  
  loading 28, 109  
  member, load 27 |
|---|---|
| U | UNMARK function 73, 86, 111  
  user exit  
  activating resources 227  
  AON messaging 225  
  CNMSAMP samples 218, 221  
  deactivating resources 227  
  EXIT01 216  
  EXIT02 216  
  EXIT03 216  
  EXIT04 216  
  EXIT05 218  
  EXIT06 220  
  EXIT07 223  
  EXIT08 225  
  EXIT09 227  
  EXIT10 230  
  EXIT11 232  
  EXIT12 232  
  EXIT13 234  
  EXIT14 236  
  EXIT15 239  
  getting resource information 218  
  inform policy post-processing 232  
  inform policy preprocessing 232  
  ipport processing 234, 239  
  NCP recovery 216  
  notification processing 230  
  overview 9  
  resource automation status 223  
  SNMP MIB polling 236  
  socket monitoring 234, 239  
  tailoring automation 213  
  threshold analysis 220  
  user-written code 118 |
| V | VTAM messages 259 |
| W | who should read this document ix  
  WKSTOPER 8  
  WTOR facility 9 |
| X | X25OPER 8 |