ViewPoint Manual

Abstract

This manual describes ViewPoint, a multifunction operations console application that allows management of a network of systems. The manual contains information on installing, configuring, and starting ViewPoint for custom applications. It also describes the concepts underlying ViewPoint operation. This manual is intended for use by operators, system or application managers, and programmers.

Product Version

ViewPoint D30

Supported Releases

This manual supports D30.00 and G05.00 and all subsequent releases until otherwise indicated in a new edition.

Part Number	Published
426801-001	July 2000

Document History

Part Number	Product Version	Published
426801-001	ViewPoint D30	July 2000
424874-001	ViewPoint D30	December 1999
133484	ViewPoint D20	April 1997

Ordering Information

For manual ordering information: domestic U.S. customers, call 1-800-243-6886; international customers, contact your local sales representative.

Document Disclaimer

Information contained in a manual is subject to change without notice. Please check with your authorized representative to make sure you have the most recent information.

Export Statement

Export of the information contained in this manual may require authorization from the U.S. Department of Commerce.

Examples

Examples and sample programs are for illustration only and may not be suited for your particular purpose. The inclusion of examples and sample programs in the documentation does not warrant, guarantee, or make any representations regarding the use or the results of the use of any examples or sample programs in any documentation. You should verify the applicability of any example or sample program before placing the software into productive use.

U.S. Government Customers

FOR U.S. GOVERNMENT CUSTOMERS REGARDING THIS DOCUMENTATION AND THE ASSOCIATED SOFTWARE:

These notices shall be marked on any reproduction of this data, in whole or in part.

NOTICE: Notwithstanding any other lease or license that may pertain to, or accompany the delivery of, this computer software, the rights of the Government regarding its use, reproduction and disclosure are as set forth in Section 52.227-19 of the FARS Computer Software—Restricted Rights clause.

RESTRICTED RIGHTS NOTICE: Use, duplication, or disclosure by the Government is subject to the restrictions as set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 52.227-7013.

RESTRICTED RIGHTS LEGEND: Use, duplication or disclosure by the Government is subject to restrictions as set forth in paragraph (b)(3)(B) of the rights in Technical Data and Computer Software clause in DAR 7-104.9(a). This computer software is submitted with “restricted rights.” Use, duplication or disclosure is subject to the restrictions as set forth in NASA FAR SUP 18-52 227-79 (April 1985) “Commercial Computer Software—Restricted Rights (April 1985).” If the contract contains the Clause at 18-52 227-74 “Rights in Data General” then the “Alternate III” clause applies.

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

ViewPoint Manual

2. Using ViewPoint (continued)

3. Definition of ViewPoint Screens

3. Definition of ViewPoint Screens (continued)

4. Process Definition Commands

4. Process Definition Commands (continued)

5. Functional Description

6. Customizing ViewPoint

6. Customizing ViewPoint (continued)

7. Installation, Configuration, and Startup

7. Installation, Configuration, and Startup (continued)

A. Server Assigns and Parameters

B. Error Messages

C. ViewPoint Event Messages

D. Sample Custom Status Server

E. Supplemental Information for D-Series Systems

Examples

Figures

Figures (continued)

Figures (continued)

Tables

Tables (continued)

What's New in This Manual

Manual Information

Abstract

This manual describes ViewPoint, a multifunction operations console application that allows management of a network of systems. The manual contains information on installing, configuring, and starting ViewPoint for custom applications. It also describes the concepts underlying ViewPoint operation. This manual is intended for use by operators, system or application managers, and programmers.

Product Version

ViewPoint D30

Supported Releases

This manual supports D30.00 and G05.00 and all subsequent releases until otherwise indicated in a new edition.

Part Number	Published
426801-001	July 2000

Document History

Part Number	Product Version	Published
426801-001	ViewPoint D30	July 2000
424874-001	ViewPoint D30	December 1999
133484	ViewPoint D20	April 1997

New and Changed Information

This is the seventh edition of the *ViewPoint Manual.* The following changes have been made to this edition of the manual.

- ViewPoint can now handle 70 Alternate Screens on G-Series systems, and 50 Alternate Screens on D-Series systems.
- The *ViewPoint Manual* has been updated to comply with the new rebadging guidelines.

About This Manual

Introduction

ViewPoint is a multiple-function application that allows you to manage a network of Compaq systems from a single terminal such as from an operator console. It provides an integrated view of the status of a group of distributed systems and the events that occur on the systems. ViewPoint is easily distributed, has fault-tolerant operation, supports multiple terminals, and can be customized by the user.

This manual discusses aspects of the ViewPoint console application that are of interest to several types of users:

- The person who uses ViewPoint to monitor and control systems and subsystems in a Compaq network. This user should be familiar with the Compaq Tandem Advanced Command Language (TACL).
- The person who installs and configures ViewPoint. This user should be familiar with configuring a Pathway application.
- The person who customizes the ViewPoint application should know how to write Pathway requesters and servers and how to write filters using the Event Management Service (EMS) filter language. This person also might need to know how to use the Subsystem Programmatic Interface (SPI).

Related Documentation

Other manuals that extend the understanding of the ViewPoint application are:

- *ViewPoint Basic Operations (Independent Study Program)* gives a detailed description of how to perform basic operations in ViewPoint.
- *Introduction to NonStop Operations Management* gives an introduction to operator tasks, including how to monitor a network with ViewPoint.
- *SPI Programming Manual* gives a detailed description of how to use the Subsystem Programmatic Interface (SPI) to write management applications.
- *EMS Manual* gives a detailed description of EMS event processing and of the EMS filter language for filtering events.
- *TACL Programming Guide* gives a detailed description of TACL.
- *DSM Template Services Manual* gives a detailed description of the template language and procedures for compiling and installing the templates.
- *Measure User's Guide* gives a description of how to use the Measure product for such functions as configuring measurements and collecting and displaying data for balancing and tuning Compaq *NonStop*™ systems.

Summary of Contents

The manual describes how to use the ViewPoint application in the following sections and appendixes:

Sections

- Section 1, Introduction to ViewPoint gives a concise overview of ViewPoint capabilities.
- Section 2, Using ViewPoint lists step-by-step procedures to help the person who monitors and controls the system to get started with using ViewPoint.
- Section 3, Definition of ViewPoint Screens describes the fields appearing on the screens and the function keys that you use with the screens.
- Section 4, Process Definition Commands describes specifications for the commands in the Define Process library, including considerations for usage.
- Section 5, Functional Description gives a conceptual overview of how ViewPoint performs the operations available to the person monitoring and controlling the systems. Introduces all terms and concepts pictorially.
- Section 6, Customizing ViewPoint describes how to add custom features to extend the ViewPoint application.
- Section 7, Installation, Configuration, and Startup explains how to install, configure, and start the ViewPoint application.

Appendixes

- Appendix A, Server Assigns and Parameters lists many of the SET SERVER ASSIGN values that can be specified in the Pathway configuration file for ViewPoint.
- Appendix B, Error Messages lists error and warning messages returned by ViewPoint.
- Appendix C, ViewPoint Event Messages lists event messages that ViewPoint reports to Event Management Service (EMS).
- Appendix D, Sample Custom Status Server shows an example of how to write a custom status server.
- Appendix E, Supplemental Information for D-Series Systems explains how to use ViewPoint on a D-series system.

In addition, the manual contains a glossary of terms used in this manual. The first time a glossary item appears in the text, the item is printed in **boldface type**.

How to Use This Manual

You are probably responsible for specific tasks within your company or site. The following table is a guideline to which sections and appendixes are most useful for your responsibilities.

Your Tasks	Read these Sections and Appendixes
Monitor network or system status and events	1, 2, and 3
Manage an application or the system	1, 2, 3, 7, and A
Customize ViewPoint (often a programmer)	1, 2-4, especially 5, 6, and 7, also appendixes A-E
Install and configure ViewPoint	1, 6, and 7
Use Define Process	4

Your Comments Invited

After using this manual, please take a moment to send us your comments. You can do this by returning a Reader Comment Card or by sending an Internet mail message.

A Reader Comment Card is located at the back of printed manuals and as a separate file on the Compaq CD Read disc. You can either FAX or mail the card to us. The FAX number and mailing address are provided on the card.

Also provided on the Reader Comment Card is an Internet mail address. When you send an Internet mail message to us, we immediately acknowledge receipt of your message. A detailed response to your message is sent as soon as possible. Be sure to include your name, company name, address, and phone number in your message. If your comments are specific to a particular manual, also include the part number and title of the manual.

Many of the improvements you see in Compaq manuals are a result of suggestions from our customers. Please take this opportunity to help us improve future manuals.

Notation Conventions

General Syntax Notation

The following list summarizes the notation conventions for syntax presentation in this manual.

UPPERCASE LETTERS. Uppercase letters indicate keywords and reserved words; enter these items exactly as shown. Items not enclosed in brackets are required. For example:

```
MAXATTACH
```

lowercase italic letters. Lowercase italic letters indicate variable items that you supply. Items not enclosed in brackets are required. For example:

```
file-name
```

[ ] Brackets. Brackets enclose optional syntax items. For example:

```
TERM [\system-name.]$terminal-name

INT[ERRUPTS]
```

A group of items enclosed in brackets is a list from which you can choose one item or none. The items in the list may be arranged either vertically, with aligned brackets on each side of the list, or horizontally, enclosed in a pair of brackets and separated by vertical lines. For example:

```
LIGHTS [ ON              ]
       [ OFF             ]
       [ SMOOTH [ num ] ]

K [ X | D ] address-1
```

{ } Braces. A group of items enclosed in braces is a list from which you are required to choose one item. The items in the list may be arranged either vertically, with aligned braces on each side of the list, or horizontally, enclosed in a pair of braces and separated by vertical lines. For example:

```
LISTOPENS PROCESS { $appl-mgr-name }
                  { $process-name  }

ALLOWSU { ON | OFF }
```

| **Vertical Line.** A vertical line separates alternatives in a horizontal list that is enclosed in brackets or braces. For example:

```
INSPECT { OFF | ON | SAVEABEND }
```

… Ellipsis. An ellipsis immediately following a pair of brackets or braces indicates that you can repeat the enclosed sequence of syntax items any number of times. For example:

```
M address-1 [ , new-value ]...

[ - ] {0|1|2|3|4|5|6|7|8|9}...
```

An ellipsis immediately following a single syntax item indicates that you can repeat that syntax item any number of times. For example:

```
"s-char..."
```

Punctuation. Parentheses, commas, semicolons, and other symbols not previously described must be entered as shown. For example:

```
error := NEXTFILENAME ( file-name ) ;

LISTOPENS SU $process-name.#su-name
```

Quotation marks around a symbol such as a bracket or brace indicate the symbol is a required character that you must enter as shown. For example:

```
"[" repetition-constant-list "]"
```

Item Spacing. Spaces shown between items are required unless one of the items is a punctuation symbol such as a parenthesis or a comma. For example:

```
CALL STEPMOM ( process-id ) ;
```

If there is no space between two items, spaces are not permitted. In the following example, there are no spaces permitted between the period and any other items:

```
$process-name.#su-name
```

Line Spacing. If the syntax of a command is too long to fit on a single line, each continuation line is indented three spaces and is separated from the preceding line by a blank line. This spacing distinguishes items in a continuation line from items in a vertical list of selections. For example:

```
ALTER [ / OUT file-spec / ] CONTROLLER

   [ , attribute-spec ]...
```

!i and !o. In procedure calls, the !i notation follows an input parameter (one that passes data to the called procedure); the !o notation follows an output parameter (one that returns data to the calling program). For example:

```
CALL CHECKRESIZESEGMENT (  segment-id                    !i
                         , error        ) ;              !o
```

!i,o. In procedure calls, the !i,o notation follows an input/output parameter (one that both passes data to the called procedure and returns data to the calling program). For example:

```
error := COMPRESSEDIT ( filenum ) ;                      !i,o
```

!i:i. In procedure calls, the !i:i notation follows an input string parameter that has a corresponding parameter specifying the length of the string in bytes. For example:

```
error := FILENAME_COMPARE_ (  filename1:length           !i:i
                            , filename2:length ) ;       !i:i
```

!o:i. In procedure calls, the !o:i notation follows an output buffer parameter that has a corresponding input parameter specifying the maximum length of the output buffer in bytes. For example:

```
error := FILE_GETINFO_ (  filenum                        !i
                        , [ filename:maxlen ] ) ;        !o:i
```

Notation for Messages

The following list summarizes the notation conventions for the presentation of displayed messages in this manual.

Nonitalic text. Nonitalic letters, numbers, and punctuation indicate text that is displayed or returned exactly as shown. For example:

```
Backup Up.
```

lowercase italic letters. Lowercase italic letters indicate variable items whose values are displayed or returned. For example:

```
p-register
```

```
process-name
```

[ ] Brackets. Brackets enclose items that are sometimes, but not always, displayed. For example:

```
Event number = number [ Subject = first-subject-value ]
```

A group of items enclosed in brackets is a list of all possible items that can be displayed, of which one or none might actually be displayed. The items in the list might be arranged either vertically, with aligned brackets on each side of the list, or horizontally, enclosed in a pair of brackets and separated by vertical lines. For example:

```
LDEV ldev [ CU %ccu | CU %... ] UP [ (cpu,chan,%ctlr,%unit) ]
```

{ } Braces. A group of items enclosed in braces is a list of all possible items that can be displayed, of which one is actually displayed. The items in the list might be arranged either vertically, with aligned braces on each side of the list, or horizontally, enclosed in a pair of braces and separated by vertical lines. For example:

```
LBU { X | Y } POWER FAIL
```

```
process-name State changed from old-objstate to objstate
{ Operator Request. }
{ Unknown.          }
```

| Vertical Line. A vertical line separates alternatives in a horizontal list that is enclosed in brackets or braces. For example:

```
Transfer status: { OK | Failed }
```

% Percent Sign. A percent sign precedes a number that is not in decimal notation. The %þnotation precedes an octal number. The %Bþnotation precedes a binary number. The %Hþnotation precedes a hexadecimal number. For example:

```
%005400
```

```
P=%p-register E=%e-register
```

Notation for Management Programming Interfaces

UPPERCASE LETTERS. Uppercase letters indicate names from definition files; enter these names exactly as shown. For example:

```
ZCOM-TKN-SUBJ-SERV
```

lowercase letters. Words in lowercase letters are words that are part of the notation, including Data Definition Language (DDL) keywords. For example:

```
token-type
```

!r. The !r notation following a token or field name indicates that the token or field is required. For example:

```
ZCOM-TKN-OBJNAME        token-type ZSPI-TYP-STRING.          !r
```

!o. The !o notation following a token or field name indicates that the token or field is optional. For example:

```
ZSPI-TKN-MANAGER        token-type ZSPI-TYP-FNAME32.         !o
```

Change Bar Notation

Change bars are used to indicate substantive differences between this edition of the manual and the preceding edition. Change bars are vertical rules placed in the right margin of changed portions of text, figures, tables, examples, and so on. Change bars highlight new or revised information. For example:

> The message types specified in the REPORT clause are different in the COBOL85 environment and the Common Run-Time Environment (CRE).
>
> The CRE has many new message types and some new message type codes for old message types. In the CRE, the message type SYSTEM includes all messages except LOGICAL-CLOSE and LOGICAL-OPEN.

1 Introduction to ViewPoint

Introduction

ViewPoint is a software product that runs on a console and is used to observe and control conditions in a Compaq system or network of systems. ViewPoint is a Compaq application of the Pathway transaction processing system. It is one of the set of products referred to as Distributed Systems Management (DSM).

Any number of network or system operators can use the application simultaneously, each viewing and controlling separate aspects of network (or system) operation through the ViewPoint screen displays and special-purpose commands. Control can occur at one terminal or system in the network or at a number of terminals on a number of systems.

You can run ViewPoint from any Compaq 6530-compatible terminal. (You should not use the Operations and Service Processor (OSP) to collect messages because it does not operate in block mode. Also, the IBM 3270 terminal is not supported.)

ViewPoint Features

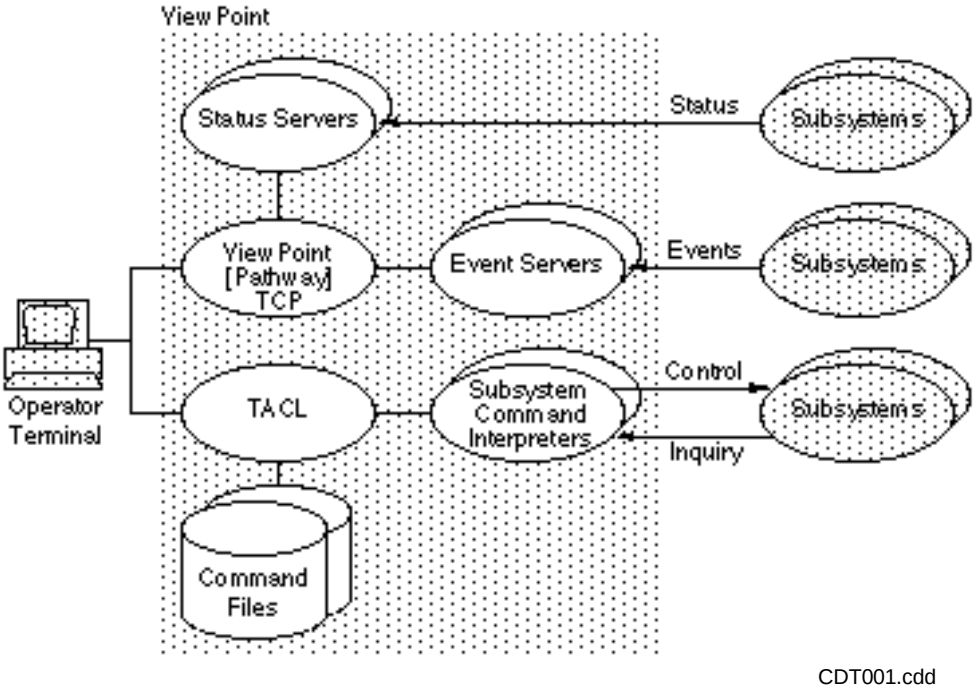
ViewPoint provides an integrated view of the systems in a distributed network and includes the following features:

- Displays the status of a variety of objects in the network on a single block-mode status screen. The status screen allows operators to monitor the availability and usage of objects such as CPUs, disks, terminals, and communications lines.
- Displays messages about current or past events occurring anywhere in the network on a set of block-mode events screens. The events screens allow operators to monitor significant occurrences or problems in the network as they occur. Critical events or events requiring immediate action are highlighted.
- Optionally, includes a conversational TACL (Compaq Tandem Advanced Computer Language) screen for interaction with local or remote command interpreters and with a Define Process library that allows concurrent sessions with multiple command interpreters from a single terminal. Operators at the TACL screen can communicate with command interpreters or other utilities, and the operators can use the Define Process commands to run a number of background processes simultaneously.
- Interfaces with high-PIN servers and processes enable the following capabilities to be configured into the product:
  - Processes can run at a high PIN if there is a high-PIN available for your system. (However, this does not include the following processes: ZVPT-EVNT-COLL, ZVPT-EVNT-NTFY, and ZVPT-EVLE-COLL).
  - ViewPoint processes can communicate with high-PIN servers.
  - High-PIN processes can create ViewPoint processes.
  - ViewPoint processes can create high-PIN processes.

- High-PIN requesters can open ViewPoint processes.
- C-series systems can support remote EMS distributors.

Also, ViewPoint can be customized to suit individual needs. Application managers can control what information is displayed on the status and event screens. Programmers can use ViewPoint as a foundation for new management applications. They can add custom screens, event-message text, event filters, and status servers and can integrate custom subsystems into the application. Figure 1-1 is a simplified overview of the ViewPoint application.

Figure 1-1. Overview of ViewPoint Application



ViewPoint Functions

The ViewPoint application has three main functions: it monitors the status of a network, it monitors events occurring in the network, and it controls or inquires about other processes in the network.

Monitoring Network Status

The ViewPoint application provides a block-mode interface for displaying network status information. ViewPoint uses system-resource information collected by the Measure subsystem and by other subsystems to provide status information about all the resources in a network. It displays this information on the Network Status Summary screen.

Network Status Summary Screen

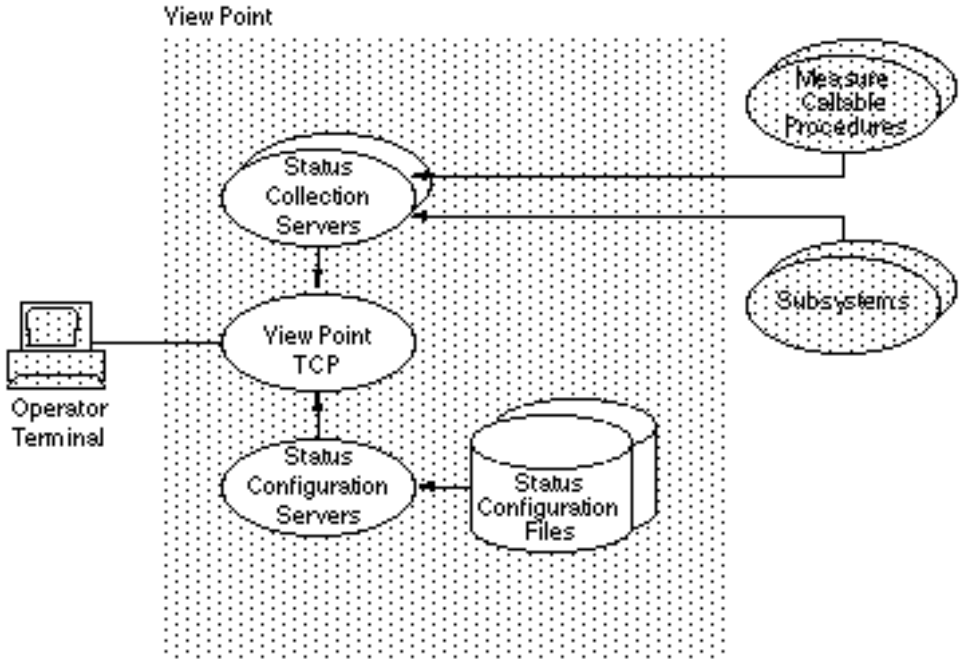
The ViewPoint operator uses the Network Status Summary screen to observe the status of items (up to 16 at a time) in a network. A status item is usually an object such as a CPU, disk, or terminal; or it might be an average, such as the average CPU usage. Each item appears on a separate line and consists of a brief description followed by a bar graph that reports on the item's availability or current usage. The screen also gives you an actual count (or percentage value) for each measured item.

Depending on the configuration, the data shown for a status item might be highlighted. Highlighting indicates that the count or percentage shown is greater (or less) than a specified value, and it usually means that the item deserves special attention.

You can save status data for any selected items in a clipboard file for subsequent printing, editing, or input to a program.

Figure 1-2 is a simplified illustration of how ViewPoint monitors network status.

Figure 1-2. Overview of Monitoring Network Status



Configuring the Network Status Screen

ViewPoint determines how to build a status display by looking at the current status configuration file. A default configuration file is supplied with the application.

However, the ViewPoint operator or application manager can use the Profile screen to specify another existing status configuration file as the current configuration file.

Two configuration screens, the Status Configuration screen and the Status Item Configuration screen, allow the operator or application manager to control what is contained in a status configuration file. When status configuration is changed on these screens, the change is recorded in the current status configuration file. ViewPoint supplies default configuration files that are shared by all users of the ViewPoint system. Thus, a default configuration file should be changed only if all users desire the change. Individual users can create configuration files for their own use.

The ViewPoint operator or application manager can use the Status Configuration screen to control how frequently item status is updated on the Network Status Summary screen. For example, the operator could specify that items be updated every 30 seconds rather than every 10 seconds (the default time period).

The ViewPoint operator or application manager can use the Status Item Configuration screen to modify or add new item descriptions. For example, the operator could specify that the description of a particular disk be highlighted when the disk is more than 60 percent busy. Or, the operator could delete the description of a disk that is no longer of interest and add a new disk to the status screen.

Monitoring Events

The ViewPoint application provides a block-mode interface for the Event Management Service (EMS). ViewPoint displays event messages collected and distributed by EMS from various subsystems throughout a system or network. The messages are displayed on three different events screens: the Primary Events screen, Alternate Events screen, and Last Events screen. Detailed explanations of the events shown on any of these screens are available on an Event Detail screen.

Primary Events Screen

Every ViewPoint operator sees the same set of event messages on the Primary Events screen. Event messages on this screen (and on other events screens) can report device malfunctions or changes in device status, or they can be requests to perform an action, such as mounting a tape. By default, this screen displays messages about events that are currently being reported at the local node of the network, including events that are being forwarded from remote nodes. The event messages on this screen are continually updated as new events occur in the system.

Alternate Events Screen

Each ViewPoint operator can see event messages configured specifically by and for that operator on the Alternate Events screen. These messages might be the same as the messages on the Primary Events screen. Usually, however, the messages on the Alternate Events screen report only on events occurring at particular subsystems in the network. This screen allows the operator to select specific events to monitor by specifying a particular EMS filter.

Last Events Screen

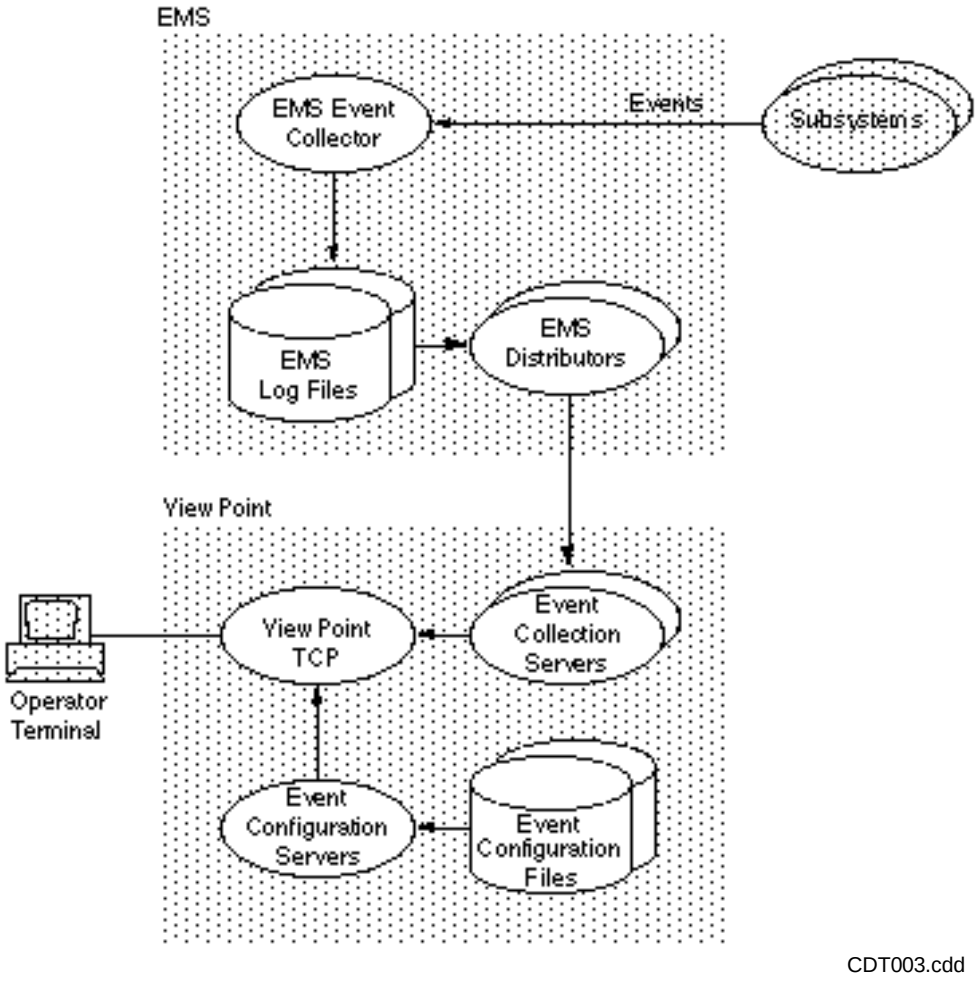
Any ViewPoint operator can display on the Last Events screen recent messages that describe events for whose subject is a particular object. The display is similar to that on the other events screens but is limited to one subject; for instance, messages about a particular device or messages originating in a particular process.

Event Detail Screen

When a ViewPoint operator needs more information about a particular event message, the operator can request the Event Detail screen. This screen is available from any of the other event monitoring screens. It displays up to five pages of event text and 99 pages of cause, effect, and recovery information for the event. It identifies the subsystem where the message originated and provides an explanation of what probably caused the event and what the operator can do about it. The cause, effect, and recovery information displayed on this screen is maintained in an event-detail database supplied with ViewPoint. A programmer (or the person managing applications at your site) can modify this database to provide site-specific explanations.

Figure 1-3 is a simplified illustration of how ViewPoint monitors events.

Figure 1-3. Overview of Monitoring Events



As with the Network Status Summary screen, the operator can save all or selected event messages in a clipboard file for subsequent printing or editing or for input to a program.

Configuring the Events Screens

ViewPoint determines how to build an event display by looking at the current **event configuration file**. A default event configuration file is supplied with the application. However, the ViewPoint operator or application manager can use the **Profile screen** to specify another existing event configuration file as the current configuration file.

Two configuration screens allow you to alter an event configuration file: the **Primary Event Configuration screen** and the **Alternate Event Configuration screen**. When you use these screens to change event configuration, the change is recorded in the current event configuration file. As a general rule, you should make your changes to a configuration file that is not the default file.

In addition to the event configuration file, event displays are controlled by filters, with each filter kept in a **filter object file**. A filter can be associated with any EMS distributor to control the event messages retrieved by ViewPoint event collection servers. Default filters are supplied with ViewPoint.

The default filter for event messages displayed on the Primary Events screen is kept in a filter object file that is specified when ViewPoint is installed. This filter cannot be changed without stopping and restarting ViewPoint. The filter for alternate events, however, can be configured during ViewPoint operation. Not only can the person monitoring ViewPoint change filters for event messages displayed on the Alternate Events screen, but this person can also pass parameters to the selected filter for dynamic control over the filtering of event messages.

The person monitoring ViewPoint or managing applications, can use the Primary Events Configuration screen to control the types of events displayed on the Primary Events screen and the frequency with which event messages are updated. For example, the Primary Events screen could be set to display only action events and critical events, rather than all events (the default specification) or to update event messages every 30 seconds rather than every 10 seconds (the default time period).

The person monitoring ViewPoint or managing applications, can use the Alternate Events Configuration screen to modify the display on the Alternate Events screen. The operator can specify a number of configuration options that control what is displayed on this screen. Changes to the Alternate Events screen do not affect the event messages displayed on the Primary Events screen. Possible configuration changes include:

- Specifying up to five EMS collectors to collect events from remote systems or EMS Alternate Collectors
- Specifying a particular filter to filter events sent to the screen
- Restricting events to those occurring at a particular system, subsystem, CPU, or process
- Specifying a particular event number or event text to be displayed
- Specifying an EMS log file to collect previous events
- Passing custom parameters to the selected filter file

Control and Inquiry

Control and inquiry functions are provided through two related parts of the ViewPoint application: the TACL screen and the Define Process library of commands.

Not all ViewPoint applications provide access to TACL and the Define Process library. Although these features are always supplied with ViewPoint, you might choose to install ViewPoint without access to TACL. You might prefer that ViewPoint run at one or more dedicated terminals or as a program under direct Pathway control. Not allowing access to TACL means more control over system security and less training for those who are monitoring, such as an operator.

TACL Screen

If available, the TACL screen provides all the existing capabilities of TACL (see the *TACL Programmer's Guide*) plus access to the Define Process library of commands. The TACL screen is part of ViewPoint, so you can use function keys on this screen to switch readily to other parts of the ViewPoint application. Conversely, you can easily switch back to TACL function, such as checking the availability of a file and running a program such as PERUSE to check on printer status.

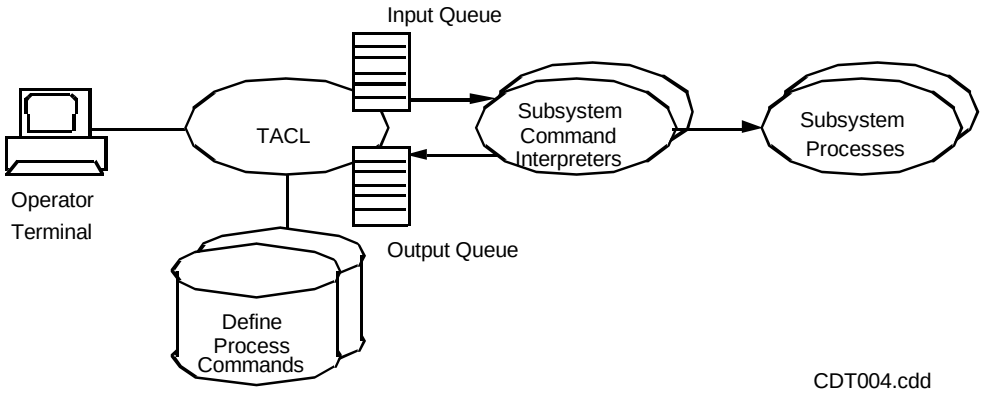
Define Process Library

The Define Process library is a library of TACL commands available through ViewPoint. These commands permit the creation of multiple subsystem control processes throughout the network. Such control processes keep running as background processes, available for access by the operator without the normal delays associated with starting a remote process. This essentially provides easy access to all available management utilities that exist in a system or network.

Define Process also enables you to create composite commands that direct commands to one or more subsystem control processes. This capability manages the communication between operators and defined background processes through queues. Commands to a process are queued in an input queue; responses from a process are queued in an output queue.

Figure 1-4 is a simplified illustration of how ViewPoint manages control and inquiry through the TACL screen.

Figure 1-4. Overview of Control and Inquiry



Customizing ViewPoint

ViewPoint can be fully customized with features tailored for a particular site or system or even a particular terminal.

If your responsibility is to customize ViewPoint, you can use one of the following procedures:

- Add custom event-message text to display messages on the Event Detail screen that reflect your own operation policies and procedures. You do this by adding text to the event-detail database with the ENABLE application generator.
- Write custom filters that determine what events are displayed on ViewPoint events screens—for instance, you could specify as critical events those events that ViewPoint does not normally consider critical. You do this with the Event Management Service (EMS) filter language.
- Add objects to the status screens and write custom server processes to provide status items not provided by ViewPoint. You do this by writing a server process in a programming language accepted by Pathway and then integrating the server into the ViewPoint application.
- Add custom commands to perform specific functions such as starting a large number of terminals or communications lines. You do this by writing TACL routines or macros.
- Add custom screens that the ViewPoint operator can request through the Extras screen. You do this by writing a Screen COBOL program unit for execution by Pathway.
- Alter the prefix text that is displayed for individual events on the events screens. You do this by setting the EVENT-PREFIX server parameter.
- Determine the options for recovering events that take effect after there is a CPU failure or ViewPoint is shut down.
- Configure ViewPoint to run with the DSM/Problem Manager (DSM/PM) product.
- Install ViewPoint as part of the Integrated Operations Console (IOC) product set. IOC consists of the following products: NetCommand, NetStatus, DSM/PM, and ViewPoint.

For more details on altering ViewPoint, see Section 6, Customizing ViewPoint.

2 Using ViewPoint

Introduction

This section is designed to help you get started quickly using ViewPoint. It covers the following topics:

- Starting ViewPoint from the TACL screen
- Navigating among the ViewPoint screens
- Monitoring the status of a network
- Monitoring events passed to your terminal
- Managing configuration files to control the content and appearance of your terminal displays
- Using ViewPoint with DSM/PM
- Using the Define Process (DP) command with TACL to start background processes of local or remote processes (such as FUP or PUP)

This section provides tutorial information on using ViewPoint to monitor status and events. It does not describe each screen in detail. For a complete description of the ViewPoint screens, see Section 3, Definition of ViewPoint Screens.

Before You Begin

Before you can start ViewPoint, the ViewPoint application must be installed, and the PATHMON process that controls ViewPoint must be started. Depending on your site, you might want to ask your site administrator to start ViewPoint or you can try starting it (see the next subsection, “Starting ViewPoint”). If ViewPoint is not installed, refer to Section 7, Installation, Configuration, and Startup for instructions on installing and configuring ViewPoint.

You can run ViewPoint on either a monochrome or color monitor. For information on running ViewPoint on a color monitor, see the subsection, “The Color Monitor” in Section 7, Installation, Configuration, and Startup.

The procedures in this manual assume that you are running ViewPoint with TACL and are using the configuration information initially supplied by Compaq. If your system manager has changed these default values or has established security restrictions, you might need to ask your system manager for additional information before starting the installation procedures.

Starting ViewPoint

To start ViewPoint on a monochrome monitor, give the following command at the TACL prompt:

```
VIEWPT [pathmon-name]
```

pathmon-name

is the name of the PATHMON process with which ViewPoint is to establish communication. The default is $ZVPT.

An example of starting ViewPoint is:

```
VIEWPT $mar
```

You will see the ViewPoint banner while ViewPoint initializes your access. When initialization is complete, you see the TACL screen and ViewPoint function keys. If you do not get the ViewPoint product banner, ViewPoint might not have been installed. See Section 7, Installation, Configuration, and Startup.

If you are not able to access ViewPoint, you might not have set your TACL #USELIST to include ViewPoint in the :UTILS directory. If this is the case, type the following fully-qualified file name at the TACL prompt:

```
:UTILS:VIEWPT [pathmon-name]
```

pathmon-name

The name of the PATHMON process with which ViewPoint is to establish communication. The default is $ZVPT. The :UTILS directory is explained in Section 7, Installation, Configuration, and Startup.

Exiting ViewPoint

To exit ViewPoint, press the following function key:

SF16

This action returns control to the TACL command interpreter. (You might have been at a TACL prompt previously, but ViewPoint was in control.) The ViewPoint processes are still active, and you can start and stop ViewPoint as many times as you need.

When you want to start ViewPoint again, at the TACL prompt, type:

```
VIEWPT [pathmon-name]
```

pathmon-name

is the name of the PATHMON process with which ViewPoint is to establish communication. The default is $ZVPT.

If you want to shut down the ViewPoint processes, first exit ViewPoint by pressing the SF16 function key, then at the TACL prompt, type:

```
VIEWPT :shutdown [pathmon-name]
```

pathmon-name

is the name of the PATHMON process with which ViewPoint is to establish communication. The default is $ZVPT.

```
VIEWPT :stopit [pathmon-name]
```

pathmon-name

is the name of the PATHMON process with which ViewPoint is to establish communication. The default is $ZVPT.

If you have not set your #USELIST to include ViewPoint, type:

```
:UTILS:VIEWPT :shutdown [pathmon-name]
```

pathmon-name

is the name of the PATHMON process with which ViewPoint is to establish communication. The default is $ZVPT.

or

```
:UTILS:VIEWPT :stopit [pathmon-name]
```

pathmon-name

is the name of the PATHMON process with which ViewPoint is to establish communication. The default is $ZVPT.

Note. If ViewPoint stops for any reason and is then restarted, ViewPoint does not display events sent while it was stopped. The person managing the ViewPoint application can get ViewPoint to display these messages by setting the configuration parameter, RECOVER-CACHE, to TRUE. See Appendix A, Server Assigns and Parameters for details.

Monitoring Status

You can monitor the status of the network with the Network Status Summary screen.

Observing Network Status

You can display information on the current usage of network resources. To observe the status of your network resources, follow these steps:

1. Press F2 to display the Network Status Summary screen.

The advice line says “Sampling” as ViewPoint collects data on the current usage of network resources. This data is then displayed and is continually updated with the most current information.

If any status line is highlighted, it means that the item is above maximum or below minimum threshold values. For information on modifying the threshold of an item, see the subsection “Changing the Item Threshold” later in this section.

The Network Status Summary screen can hold up to 16 status items. When more than 16 items are received, the display automatically continues on a new page.

2. To see an item displayed on a previous or later page, press PREV PAGE and NEXT PAGE (on some terminals, PG UP and PG DN). To select the first page, press SHIFT-PREV PAGE (or ALT-PG UP). Alternatively, you can type the page number in the **option line**, then press F2. (See Section 3, Definition of ViewPoint Screens for the location of the option line.)

   If you want to select items from the screen (either to save to a file or to print), or if you want to change the display in any way, you should stop the display of new status items by freezing the screen. The next subsection explains how to freeze a screen.

Freezing and Thawing the Screen

You can freeze the Network Status Summary screen when you want to select items from the screen that you want to print, to add or delete items, or to change the configuration of the screen in any way. Freezing the screen ensures that you capture only those items in which you are interested and prevents new items from being displayed. To freeze and thaw the screen, follow these steps:

1. Press F2 to display the Network Status Summary screen.
2. Select a page you want to freeze.
3. Press F8 to freeze the screen.

   Note that there are no more updates, and the word FROZEN blinks in the upper right corner of the screen. You can select an item by pressing Tab, Up Arrow or Down Arrow, Return, or the space bar.

4. When you are ready to start collecting and displaying status items once again, press SF8 to thaw the screen.

Printing Status Information

You can copy the contents of any screen to a file. Then, you can print the file.

To copy the contents of a screen to a file and print it, follow these steps:

1. Press F2 to display the Network Status Summary screen.
2. Select a page you want to print.
3. Press F8 to freeze the screen and select about half of the lines on the screen by typing any character in column 1 of those lines.

4. Press F10 to copy the selected lines to your **clipboard file** (ZZVPCLIP) in your default logon subvolume.

   If the clipboard file did not exist previously, it is created now. “Clipping” is displayed in the advice line. When the copying is done, “Lines clipped” is displayed.

5. Specify a printer. Move the cursor to the option line and type:

   ```
   $S.#printer
   ```

   where *printer* is the name of a printer on your system.

6. Press F9 to print the file to the spooler file you specified.

   When the printing is done, “Screen printed” is displayed.

7. Press SF8 to thaw the screen and continue displaying network status.

8. To verify the contents of the ZZVPCLIP file, follow these steps:

   a. Press F1 to go to the TACL screen.

   b. To display a list of the files, type FILES.

      Note that the ZZVPCLIP file is now present. ZZVPCLIP is created in your logon subvolume, so if you are working in another subvolume, request FILES *logon-subvolume*.

   c. Type > TEDIT ZZVPCLIP to observe the contents of the file.

   d. Press SF16 to exit TEDIT.

   e. Type > PURGE ZZVPCLIP to delete the test edit file.

Changing the Status Interval

If you want to display the status information more or less frequently, you can change the time between updates of the status display.

To change the time between updates, follow these steps:

1. Press F1 to display the TACL screen, if it is not already selected.

2. Press F2 to display the Network Status Summary screen.

3. Move the cursor to the option line and press F12 to display the Status Configuration screen.

   Note the sampling interval and write it down for future use.

4. Change the sampling interval to something significantly different, such as half or double by typing over the original value.

5. Press F12 to confirm your change and return to the Network Status Summary screen.

   Observe your new frequency of updates.

6. Move the cursor to the option line and press F12 to return to the Status Configuration screen.
7. Restore the original update interval noted in Step 3. (The interval value affects only this terminal.)
8. Press F12 to confirm the change and return to the Network Status Summary screen.

Changing the Item Threshold

You can cause a status item to be highlighted on the screen whenever its computed usage value falls above or below specified threshold values.

To change the upper threshold value of an item, follow these steps:

1. Press F2 to display the Network Status Summary screen.
2. Press F8 to freeze the screen.
3. Move the cursor to one **status item** (not a highlighted one).

   Make a note of the value and total for that item; for example, 24/100.
4. Press F12 to display the Status Item Configuration screen.

   Make a note of the existing value for the upper threshold. Then change the upper threshold to some significantly lower value. This ensures that the threshold is exceeded when sampling resumes later.

   Also, if the field called Enable Upper Threshold does not have a Y entered, type Y in that field.
5. Press F12 to confirm the change and return to the Network Status Summary screen.
6. Press SF8 to thaw the screen.

   Note that the entire line for the selected item becomes highlighted whenever the new threshold value you set in the preceding steps is exceeded.
7. Press F12 to display the Status Item Configuration screen and restore the original value of the upper threshold, as noted in Step 5.
8. Press F12 to confirm the change and return to the Network Status Summary screen.

Changing the Item Order

You can change the order in which status items are displayed with the Status Item Configuration screen. For example, suppose you want to see the most critical items displayed at the top of the screen. To change the order of the items, you must first delete the items and then move them to a new location on the screen.

To change the order of the status items, follow these steps:

1. Press F2 to display the Network Status Summary screen.
2. Press F8 to freeze the screen.

3. Move the cursor to an item you want to move. Select one that is easy to identify after it appears in a new position.
4. Press F6 to delete the item from the display; it is saved in memory to allow you to move it to another location.
5. Move the cursor to the new desired location and press F7 to display the item at the new location. Any item currently in that location is bumped down when you add the deleted item.

   After pressing F7, the Status Item Configuration screen is displayed so you can confirm this new configuration.
6. Press F12 to confirm it.
7. Press F16 to return to the Network Status Summary screen.

   Note that the selected item is in its new location.
8. Restore the item to its original location.
9. Press F1 to return to the TACL screen.

Monitoring Events

You can monitor event messages using one of following event screens:

Primary Events	For displaying the same event messages at all ViewPoint terminals; those events of interest to all operators at a site
Alternate Events	For displaying selected event messages at particular terminals; those events of interest only to individual operators, not to all operators
Last Events	For displaying the most recent event messages that concern a particular subject
Event Detail	For displaying event text associated with a particular event

Observing Primary Events

You can observe primary event messages occurring in the system with the Primary Events screen.

To observe these messages, follow these steps:

1. Press F1 to display the TACL screen, if it is not already selected.
2. Press F3 to display the Primary Events screen. This screen shows a list of event messages.

   Critical events are displayed at full brightness (or in a distinct color if a color monitor is used). Action events are displayed in inverse video. All other events are displayed at half brightness. Note that the most recent critical or action event is also

displayed on line 25 of your terminal, and the terminal's bell tone sounds each time a new message is displayed in this line.

Line 3 tells you the name of the **event collector** process currently being used, the name of the current **filter**, and the number of action and critical events on this display.

Notice that the display for each event gives the time of the event, the system node name, and the event-message text (the first 62 characters).

3. This is the end page, as indicated in the upper-right corner of this screen. Search through the various pages of earlier events.
   - To observe earlier events, press PREV PAGE (or PG UP on some terminals).
   - When the current page is not the last page, press NEXT PAGE (or PG DN on some terminals) to observe later events.
   - To select the first page, press SHIFT-PREV PAGE (or ALT-PG UP). Alternatively, you can type the page number in the option line and then press F3.
   - To select the END page, press SHIFT-NEXT PAGE or by selecting a page number of 20 (or greater). Notice that only the END page gets updated as new event messages arrive.
4. Press F1 to return to the TACL screen.

If you want, you can print this screen by following the same steps used to print status information. For more details on printing, see "Printing Status Information" earlier in this section.

Changing the Delay Between Updates

You can change the amount of time between updates of events displayed on the Primary Events screen with the Primary Events Configuration screen. For example, suppose you want a longer or shorter interval between the updates to the Primary Events screen.

To change the delay between updates, follow these steps:

1. Press F1 to display the TACL screen, if it is not already selected.
2. Press F3 to display the Primary Events screen.
3. Press F12 to display the Primary Event Configuration screen.

   Note the current value in the field called Delay Between Updates. This is a minimum time, given in seconds. If events come at longer intervals, the screen is updated only when the event message arrives; if events come at shorter intervals, the screen is updated when the specified delay time expires.
4. Write down the current delay value and keep it for later. Enter some other value in this field. If the current value is fairly long, such as 15 or 20 seconds, change it to 1 second; if the current value is very short, such as 0 or 1 second, change it to 15 seconds.

5. Press F12 to confirm the change and return to the Primary Events screen.

   Notice the difference in the timing of updates. If you changed the value to 0, the updates are more random, occurring as soon as each new event message arrives. If you changed the value to 15, the updates are at least 15 seconds apart.

6. Restore the original delay.

   Press F12 to display the Primary Event Configuration screen and restore the original value (the number you wrote in Step 4).

7. Press F12 to confirm the change and return to the Primary Events screen.

8. Press F1 to return to the TACL screen.

Restricting the Events Displayed

You can restrict the types of events (action, critical, or normal) that are displayed on the screen with the Primary Event Configuration screen. Any combination of these events can be displayed. In the case of action and critical events, you can specify that only outstanding or acknowledged events be displayed. For example, suppose you want to see messages only for outstanding critical events.

To display messages for outstanding critical events only, follow these steps:

1. Press F1 to display the TACL screen, if it is not already selected.

2. Press F3 to display the Primary Events screen.

3. Press F12 to display the Primary Event Configuration screen and restrict the messages to only outstanding critical events.

   a. Write down the selections for each of the options so you can restore the configuration later. An X under one of the columns (All, Outstanding, Acknowledged, or None) means it is selected.

   b. Type an X under the column None for the Normal and Action options. Enter an X under the column Outstanding for the Critical option.

   c. Erase any X appearing in the other columns.

   d. Press F12 to confirm the changes. Then, return to the Primary Events screen.

      You now see messages for only outstanding critical events on your screen (all inverse-video messages or green on a color screen).

4. Press F12 to return to the Primary Event Configuration screen and restore the options to their original selection (selection you wrote in Step 3a above).

5. Press F1 to return to the TACL screen.

Displaying Message Text

When you want to know more about a particular event, you can display message text associated with the event using the Event Detail screen.

To display message text associated with an event, follow these steps:

1. Press F1 to display the TACL screen, if it is not already selected.
2. Press F3 to display the Primary Events screen.
3. Press F8 to freeze the screen. Then, select 3 or 4 event messages that appear truncated (where the text is more than the 62-character limit). Do this by typing any character next to the message.
4. Press F11 to display the Event Detail screen.

   Detail is shown for the first message you select.

   The first line shows the date and time of the event, the process that generated it, the subsystem identification number, and the event number.

   The third line contains the beginning of the full text of the message. Following this, (depending on the messages you selected) lines are provided for Cause, Effect, and Recovery text.

   To print this screen, press F9. To copy selected lines to your clipboard file, press F10. For details, see the steps under “Printing Status Information” earlier in this section.
5. Press F11 to view the event detail for the next event that you selected.
6. Press F11 one or more times, depending on how many events you selected. When you press F11 after viewing the final selected event, the display returns to the events screen. (You can also return at any time by pressing F16.)

Observing Alternate Events

You can display specific event messages with the Alternate Events screen. Depending on how your terminal is configured, the alternate-event messages might be identical to the event messages shown on the Primary Events screen.

Even if the alternate-events display is the same as the primary-events display, you still might need to request the Alternate Events screen. A number of functions are available only through the Alternate Events screen and its configuration screen. For instance, you use the Alternate Events and the Alternate Events Configuration screens to customize events for your particular terminal (as described in the steps under “Creating Configuration Files,” later in this section).

You also use the Alternate Events screen to see messages describing historical events—that is, events that occurred at a particular point in time, as shown in the next subsection “Observing Historical Events.”

Note. The maximum number of Alternate Event screens that can be configured on a single ViewPoint installation on a G-Series system is 70. The maximum number of Alternate Event screens that can be configured on a D-Series system installation is 50.

Observing Historical Events

You can see messages for events that occurred at an earlier date or time with the Alternate Event Configuration screen. For example, suppose you want to see messages for events that occurred since 1:00 p.m., yesterday.

To see the messages for these earlier events, follow these steps:

1. Press F1 to display the TACL screen, if it is not already selected.
2. Press SF3 to request the Alternate Events screen.

   Depending on how this display was last configured, you might see an events display that is similar to an alternate-events display you saw previously.
3. Press F12 to display the Alternate Event Configuration screen.

   Observe the After field and make a note of the current date and time values in this field.
4. Move your cursor to the After field, and enter yesterday's date with the year first, month next, and day last.

   An example is 92 for the year, 1992; 04 for the month of the year; and 02 for the day in the month. An example for time is 13:00 (which is 1:00 p.m based on 24-hour clock time). The field would look like the following:

   ```
   After: 92-04-02   13:00:00
   ```
5. Press F12 to confirm the configuration and to return to the Alternate Events screen. You are now in **historical mode**.

   At first, the Alternate Events display might be empty except for the message "Reading event log at tt:mm...", which is displayed in the advice line. When the Event Management Service (EMS) has located yesterday's information, it begins to return event messages, starting from the time you requested.

   Notice the dates and times of the messages. These are historical messages. Additional event messages are displayed as they are obtained by EMS. The update rate is determined by the Delay Between Updates field in the Alternate Event Configuration screen.
6. Search through the various pages of earlier events. Note that this is the end page, as indicated in the upper-right corner of this screen.

   - To observe earlier events, press PREV PAGE (or PG UP on some terminals).
   - When the current page is not the last page, press NEXT PAGE (or PG DN on some terminals) to observe later events.
   - To select the first page, press SHIFT-PREV PAGE (or ALT-PG UP). Alternatively, you can type the page number in the option line and then press F3.

- To select the END page, press SHIFT-NEXT PAGE or by selecting a page number of 20 (or greater). Notice that only the END page gets updated as new event messages arrive.

Note. The **event cache**, which contains the alternate event messages, retains its contents if you go to some other screen (such as Primary Events) and is accessible when you return. The cache is cleared only when you exit from ViewPoint or if you alter the configuration for alternate events.

---

7. Press F12 to display the Alternate Event Configuration screen and restore the original time and date values, as noted in Step 3.
8. Press F12 again to confirm the change and return to the TACL screen.

Observing Events by Subject

You can view recent event messages that have a common subject name with the Last Events screen. For example, you might want to view messages that relate to a particular logical device or subsystem.

Each event message can have one or more unique subject names. You can determine what the subject name of an event message is, if you don't know it, by selecting it on the Primary or Alternate Events screen and then displaying the Last Events screen to see the subject name associated with it. Some event messages have no subject names associated with them. When this is the case, the message "Unsuitable subject" appears on your screen. For information on determining the subject name of an event, see the description of the Subject field for the Last Events screen in Section 3, Definition of ViewPoint Screens.

To view messages associated with a particular subject, follow these steps:

1. Type the subject name on the option line.

   You need not be concerned with case when specifying the subject name; ViewPoint converts the subject name to uppercase letters before using it.
2. Press F4 to display the Last Events screen.

To view the event messages that have a common subject name, follow these steps:

1. Press F1 to display the TACL screen, if it is not already selected.
2. Press F3 to display the Primary Events screen.
3. Press F8 to freeze the screen.
4. Move the cursor to the event you are interested in.
5. Press F4 to display the Last Events screen. This screen has only one page containing 10 event messages, all on the selected subject.

   Notice the subject heading at the top of the screen. This gives you the subject name of the events you are examining. Write down this name so you can use it to select that subject later.

6. Press F3 to display the Primary Events screen.
7. Move the cursor to the option line and enter the subject you noted down from the Last Events screen in Step 5.
8. Press F4 to display the Last Events screen by subject name. The display should be the same as before, except for any new events issued since then.
9. Press F1 to display the TACL screen.

Managing Configuration

Configuration means the set of specifications that determine how status and events are displayed at your terminal. Using various configurations, you can customize the appearance and content of the information displayed on your status and events screens.

You manage configurations using the Configuration screens and the Profile screen in the following ways:

Event or Status Configuration Screen:	Specify alternate configurations as you did in previous exercises when you changed how events are displayed (see the subsection “Restricting the Events Displayed”) or when you changed the threshold of a status item (see the subsection “Changing the Item Threshold”). When you enter changes to these configuration screens, the changes are saved in a configuration file.
Profile Screen:	To name the configuration files in which to save alternate configurations.
Profile Screen:	To select alternate configuration files.

Note. Note that only two configuration files are active at any time: one for events and the other for status. However, you can create any number of other configuration files, each with an alternate configuration. See the next subsection, “Creating Configuration Files.”

ViewPoint provides these two default configuration files:

- STATDFLT, for status configuration
- EVNTDFLT, for event configuration

If you are managing applications, you can create other default files (for example, ZZVPSTAT and ZZVPEVNT) on the default logon subvolume or, for ViewPoint without TACL, on the PATHMON user ID logon default subvolume. If the ZZVPSTAT and ZZVPEVNT files exist, ViewPoint uses these as the default configuration files; otherwise, ViewPoint uses STATDFLT and EVNTDFLT.

The following procedures, “Creating Configuration Files” and “Choosing an Alternate Configuration,” show how to use the Profile screen together with the configuration screens to define and select alternate configurations.

Creating Configuration Files

You can create any number of configuration files, each with an alternate configuration. Having a set of configuration files allows you to switch easily from one configuration to another.

In the following steps, you create a new configuration file for the Network Status Summary screen and a new configuration file for the Alternate Events screen.

△ **Caution.** Be careful when making changes to existing configuration files, particularly ones that you do not want altered; for example, the default files provided with your application. Whenever you change a configuration on one of the configuration screens, this change is immediately recorded in whichever configuration file is currently active. To make sure you do not alter an existing configuration file, create a new configuration first and then make your changes as described below.

To create the new configuration files, follow these steps:

1. Press F1 to display the TACL screen, if it is not already selected.
2. Press F14 to display the Profile screen. This screen displays the names of the current event configuration and status configuration files, probably the default file names STATDFLT and EVNTDFLT (although they might be ZZVPSTAT and ZZVPEVNT). These names are qualified by the names of your current system, volume, and subvolume.
3. Change the name of the status configuration file to a new name, such as MYSTAT. (Like all file names, it must be 8 characters or less.)
4. Press F14 to make the name change.
5. Press F14 again to create the new configuration file, MYSTAT. This is a new configuration file, so ViewPoint asks you to press F14 again in order to create the file.
6. Specify the status configuration to be associated with MYSTAT. To do this, you must add the items whose status you want to display.
    a. Press F2 to display the Network Status Summary screen. Notice that the screen header now shows MYSTAT as the Status File, but that there are no status items displayed on the screen. You can now add the status items you want displayed when you use MYSTAT as your configuration file.
    b. Press F7 to add a status item. When you press F7, ViewPoint immediately displays a Status Item Configuration screen on which all fields are blank so you can add new items. Rather than entering item information on a blank screen, you can scan through templates of possible display types (such as CPU-BUSY and DISK-BUSY). When you find the type you want to add, all you need do is enter specific information in the template.
    c. Press F13 to request the first display type, which is CPU-BUSY.
    d. Press SF13 to scan the display types. Keep pressing SF13 until the screen shows the template for DISK-BUSY. To see the status of the $SYSTEM disk,

enter $SYSTEM in the Object Name field. (You should also type $SYSTEM after Disk Busy in the Item Description field so that the name of the disk appears on the screen.)

e. Make any other changes to the template that you want.

f. Press F12 to confirm the configuration change. You can continue to add new items; for instance, add all the disks on your system. ViewPoint continues to display the Status Item Configuration screen until you request a different screen.

7. Press F16 to return to the Network Status Summary screen and observe the new items you have added. This is the display shown whenever you request MYSTAT on the Profile screen as your status configuration file.

   You might use the alternate configuration file, MYSTAT, on the same terminal where you display events on an Alternate Events screen. An Alternate Events screen is particularly well suited to alternate configurations because you can change configurations on that screen without affecting the Primary Events screen.

8. Press F14 to request the Profile screen, and change the name of the event configuration file to a new name, MYEVNT.

9. Press F14 to make the name change.

10. Press F14 again to create the new configuration file, MYEVNT. This is a new configuration file, so ViewPoint asks you to press F14 again in order to create the file.

11. Define the configuration for MYEVNT. To do this, you indicate what events to display and how you want them displayed.

    a. Press SF3 to request the Alternate Events screen. The screen is blank except for the headers and a message on the advice line that asks you to press F12 to configure the screen.

    b. Press F12 to request the Alternate Event Configuration screen.

    c. Restrict the Alternate Events screen to display all action events that occurred since 5:00 yesterday evening. Then put yesterday's date (in the year/month/day format) and the time (17:00:00) in the After field.

    d. Press F12 to confirm the configuration and return to the Alternate Events screen.

       It is possible that no events are displayed on the screen; notice the number of Action Events displayed at the top of the screen. If it is zero, no events are displayed.

12. Press F1 to return to the TACL screen.

Note. You might also use the Alternate Events Configuration screen to specify remote collectors or log files on other systems from which events are to be collected (instead of using remote distributors); to specify a particular filter to control the events passed to the Alternate Events screen; or to pass information to a selected filter to further customize the filtered events.

Refer to the description of the Event Configuration screen in Section 3, Definition of ViewPoint Screens for detailed information on other configuration changes you can make.

Choosing an Alternate Configuration

The only way you can choose an alternate configuration file is through the Profile screen. You (or the person managing applications at your site) can create any number of configuration files for alternate configurations. Once the alternate configurations are associated with particular configuration files, you simply go to the Profile screen to select the file with the configuration you want.

To choose an alternate configuration file, follow these steps:

1. From the TACL screen, press F14 to request the Profile screen. The screen displays the current configuration files—in this case, probably the files MYSTAT and MYEVNT. If you want to select a different configuration, change the current file names to different file names. At this point, return to the default configurations by changing the names to STATDFLT and EVNTDFLT (or ZZVPSTAT and ZZVPEVNT).
2. Press F14 to confirm each name change.
3. Press F2 to request the Network Status Summary screen or the Primary Events screen, and observe the changed displays.
4. Press F1 to return to the TACL screen.

Using DSM/PM

ViewPoint provides an interface to the DSM/Problem Manager (DSM/PM) application. When ViewPoint is configured to run with DSM/PM, a user with a valid DSM/PM user name can do the following:

- Log on to DSM/PM from any ViewPoint event screen and perform any DSM/PM functions.
- Send an event that is marked on a ViewPoint event screen to DSM/PM as a created problem requiring additional information.
- Send events that are marked on a ViewPoint event screen to DSM/PM for problem submission.
- Send multiple events that are marked on a ViewPoint event screen to DSM/PM as information records to be subsequently classified as problems.

Accessing DSM/PM from ViewPoint

There are different ways of accessing DSM/PM from ViewPoint. The method you choose depends on the task you are trying to perform. Each method, however, involves one of two function keys: F5 or SF5. The different methods are summarized as follows:

- To move to a specific DSM/PM screen, type the name of the screen you want to access (or a valid alias for it) on the ViewPoint option line and then press F5 or SF5.
- Press SF5 to send an event marked on a ViewPoint event screen to DSM/PM as a created problem requiring additional information.
- Press F5 to send events marked on a ViewPoint event screen to DSM/PM for problem submission.
- Press SF5 to send multiple events marked on a ViewPoint event screen to DSM/PM as information records. You must subsequently classify and submit each problem.

Note. Your operating system user name and password can be defined in DSM/PM so that logging on to DSM/PM from ViewPoint is automatic. When going from ViewPoint to DSM/PM, you will not see the Logon screen as you normally would if you were accessing DSM/PM as a stand-alone application. If your user name is not defined, the DSM/PM Logon screen is displayed and you must enter your user name and password. See your DSM/PM system administrator about defining your user name and password in DSM/PM, if it is not defined already.

Adding ViewPoint Events to DSM/PM

To add an event to DSM/PM or to add and submit an event in the same step, a problem type for the event must be defined in DSM/PM. If a problem type does not exist for a particular event and you try to add that event to DSM/PM, ViewPoint displays an error message that indicates ViewPoint cannot add the event.

Adding an Event Without Submitting It

You can mark one or more events on the ViewPoint Primary, Alternate, or Last Events screen and then display the event on the DSM/PM Create Problem screen without submitting it.

To add an event from the ViewPoint Primary Event screen to DSM/PM without submitting it, follow these steps:

1. From the TACL screen, press F3 to display the Primary Event screen.
2. Select the event page that contains the event you want to add to DSM/PM.
3. Press F8 to freeze the screen.
4. Mark the event you want to add by typing any character next to the event.
5. Press SF5 to display the event on the DSM/PM Create Problem screen.

The object name associated with the event you marked is displayed as the problem subject on the screen. DSM/PM does not automatically submit the problem, but is waiting for you to take further action.

6. When you are ready, follow the DSM/PM instructions for creating and submitting the record.
7. Press F16 to return to the ViewPoint Primary Event screen.

   At this point you are still logged on to DSM/PM.
8. When you are ready to start collecting and displaying events once again, press SF8 to thaw the screen.

Submitting Events as Problems or as Problem Information Records

You can mark multiple events on the ViewPoint Primary, Alternate, or Last Events screen and then submit them to DSM/PM as problems or problem information records.

To submit the events from the ViewPoint Primary Event screen to DSM/PM as problems or problem information records, follow these steps:

1. From the TACL screen, press F3 to display the Primary Event screen.
2. Select the event page that contains the events you want to submit to DSM/PM.
3. Press F8 to freeze the screen.
4. Mark the events you want to submit by typing any character next to them.
5. Do you want to submit the events as problem information records or as problems?
   - If you are submitting the events as problem information records, press SF5.

     The Create Problem screen appears with the marked events appearing as problem information records at the bottom of the screen. You can create problems using all the functions available on the screen.
   - If you are submitting the events as problems, press F5.

     The events are automatically submitted as problems to DSM/PM. The DSM/PM screen is not displayed; instead, you continue to view the Primary Event screen.
6. Return to the ViewPoint Primary Event screen, if it is not displayed already, by pressing F16.

   At this point you are still logged on to DSM/PM.
7. When you are ready to start collecting and displaying events once again, press SF8 to thaw the screen.

Using Define Process

From ViewPoint you can enter any Define Process command. The Define Process commands allow you to start up to 100 background processes (such as PUP, FUP, and SPOOLCOM) without having to exit ViewPoint. You do not have to be within ViewPoint to enter the Define Process commands, but you must have access to the TACL process.

For reference information on the Define Process commands, refer to Section 4, Process Definition Commands.

Using Define Process Commands

Suppose you are in ViewPoint and decide you want to use Define Process to run the PERUSE and FUP processes.

To use Define Process commands, the TACL process that you are using must be named. You can use the STATUS *,TERM command on the TACL screen to check whether your TACL process is named; if it is, you can proceed with these exercises.

To use the Define Process commands from within ViewPoint, follow these steps:

1. Press F1 to go to the TACL screen.
2. Type the TACL command, DP, to put the Define Process directory in the TACL process #USELIST. Type the following:

   ```
   DP
   ```

   or if you need the fully-qualified name, type:

   ```
   :UTILS:DP
   ```

3. Run the PERUSE and FUP processes.

   a. Type the following DP command to create a PERUSE process in your local system:

      ```
      DP PERUSE
      ```

      You will see the “pstart:” message, which confirms that your defined process is now started and running.

   b. Type the following DP command to start another defined process:

      ```
      DP FUP /PNAME FP/
      ```

      The symbolic name FP is recommended so that you can easily recognize the background process you have entered.

      Both defined processes are now running, ready to accept commands from you.

   c. Type the command PINFO to get basic information about your defined processes.

Note that the name for the PERUSE defined process is the same as its program name. However, the defined process, FUP, is different; its name is FP.

The input queues (for queuing input commands) are shown to be empty, as are the output queues (for receiving output information from the processes). Both processes are ready to accept commands.

d. Type the command PERUSE.

Note that the TACL history number (enclosed in hyphens) precedes the PERUSE prompt (_) and keeps increasing as you continue to issue commands to your defined process.

e. Press BREAK to return to TACL.

The PERUSE process keeps running and remains ready for commands, as shown in the next step.

f. Type > PINFO.

Note that PERUSE is still ready.

g. Type > FP to access your defined FUP process by using its logical name.

The FUP prompt is a hyphen; thus two hyphens follow the history number.

h. Type the FUP command INFO to list the files in your subvolume.

If your subvolume has many files, try this command again and be ready to press the BREAK key midway through the listing. BREAK returns you to TACL, while leaving some output from the process in its output queue.

i. Press BREAK to return to TACL.

j. Type > PINFO and observe the output queue.

The output queue is no longer empty. It might have several text lines if there was information left to display when you pressed the BREAK key.

k. Type > PSHOW FP to examine the contents of the queue.

The remaining lines in the output queue now list out. PSHOW, however, does not empty the queue, but merely shows its contents.

4. Type > POUT FP. The same lines list out again, but this time the queue is emptied (as another PINFO would show).

5. Do you want to access a remote process?

If yes, follow the next procedure, beginning with Step 4.

If no, type > UNDP * to stop processes defined with DP.

This command removes the definitions of the two processes you defined.

6. Press SF16 to exit.

Using Remote Processes

The following instructions describe how you use remote processes. If you follow all the following steps in sequence, your ViewPoint files remain in your logon subvolume for future use and all temporary files and processes are eliminated.

Note. These steps can be performed only if you have access to some other node of a Compaq network, so you must have remote passwords established on both systems—your local one and the remote one. Instructions for establishing remote passwords are in the *Guardian User's Guide*.

---

1. Perform Steps 1 through 4 from the previous subsection, "Using Define Process Commands."
2. Type > DP \REMO.TACL /PNAME A/.

   This starts a remote TACL process named A on system node \REMO.
3. Type > A to go to the remote TACL.

   Note that the remote TACL has its own history number, that follows the local TACL history number.
4. Type > STATUS *,PROG $SYSTEM.SYSTEM.PATHMON

   From the information displayed, pick a valid PATHMON name (for example, $PTMI) for use in Step 7 and press [CTRL-Y].
5. Return to your local TACL.
6. Type the following to observe information about the remote TACL process you have started:

   ```
   > PINFO /DETAIL/ A
   ```

   Note that the process name in the PINFO display is prefixed with the name of the system node where the process is running.
7. Type > DP \REMO.PATHCOM /PNAME MYREMOTEPATHMON1/ \REMO.$PTMI to define a second remote process in that node—a PATHMON process.

   The example uses a long logical name; these names can be 1 to 16 characters.
8. Type > PINFO /DETAIL/ MYREMOTEPATHMON1 to observe information about this new remote process.

   Note that, with the long names, three lines are needed to show the detailed information for this process.

   You now have four processes defined: two local and two remote. Try going to each one of them.

   a. Type PERUSE. You are now in your local PERUSE process.

   b. Press BREAK to return to TACL.

c. Type FUP. You are now in your local FUP process.

d. Press BREAK to return to TACL.

e. Type > A. You are now in your remote TACL process.

f. Press BREAK to return to your local TACL.

g. Type >MYREMOTEPATHMON1

You are now in your remote PATHMON.

h. Press BREAK to return to your local TACL.

9. Type > UNDP * to remove the definitions of the defined processes. Press SF16 to exit.

3 Definition of ViewPoint Screens

Introduction

This section describes the screens provided by the ViewPoint application. It also defines the operation of the function keys for moving around the screens and for specific functions on each of the screens.

The screens are described in alphabetic order. They are:

- Events screen (for Primary Events screen and Alternate Events screen)
- Event Configuration screen (for Primary Event Configuration screen and Alternate Event Configuration screen)
- Event Detail screen
- Network Status Summary screen
- Profile screen
- Status Configuration screen
- TACL screen

Note. Since the Primary Events screen and the Alternate Events screen are identical, they are described as a single events screen. Also, because the Primary Event Configuration and the Alternate Event Configuration screens are almost identical—the Alternate Event Configuration screen provides a second page for additional configuration—these screens are described as a single Event Configuration screen.

The Extras screen is not listed because it is accessible only if your installation includes the custom programs that make use of this feature. If such programs exist, your application manager provides their documentation.

General Information About the Screens

You communicate with ViewPoint through your terminal, which might have either a monochrome or color monitor. The terminal screen displays 24 lines of 80 characters each, with a specially-addressed twenty-fifth line reserved for reporting the newest **outstanding event** and a running total of outstanding events. Typically, an audible warning tone (beep) sounds each time new information is written to the twenty-fifth line. This line is updated for only those screens that are themselves updated periodically (the events and status screens), as long as those screens are not frozen.

Individually-addressable display fields support inverse, half-bright, and blinking video attributes. Some of these fields cannot be altered; others are entry fields that permit (or require) you to make or change appropriate entries.

ViewPoint operates the terminal in one of two different modes: block mode or conversational mode. Conversational mode is used only for the TACL screen and is indicated by the legend TTY-FDX as the last information field on the twenty-fifth line.

All other screens use block mode, which is indicated by the word BLOCK in the same position on the twenty-fifth line. In conversational mode, characters are transmitted to the controlling process as they are typed. In block mode, information is transmitted as complete messages or blocks of data—typically when you press an appropriate function key.

Function Keys

Table 3-1 provides a summary of the ViewPoint function keys.

Table 3-1. Summary of Function-Key Assignments *(page 1 of 3)*

Key	Function	Description
F1	TACL	Displays the TACL screen and activates your TACL process.
		The interactive field of the display is initially blank except for the initial TACL prompt. You can enter a command on the option line prior to using F1; it is executed by TACL and the results displayed on the TACL screen.
		Note: This screen is not available on all ViewPoint terminals.
F2	Status	Displays the Network Status Summary screen and the first page of status.
F3	Primary Events	Displays the Events screen for primary events and the most recent page of messages in the primary-events cache.
SF3	Alternate Events	Displays the Events screen for alternate events and the most recent page of messages in the alternate-events cache.
F4	Last Events	Displays the Last Events screen and the most recent page of messages for the subject you have selected.
		Does this by positioning the cursor on an event line on the Primary or Alternate Events screen, or by entering the event name on the option line.
F5	DSM/PM	Displays the specified DSM/PM screen after you type the name of the DSM/PM screen (or any valid alias for it) on the option line of any ViewPoint screen.
		Submits event(s) previously marked on the Primary, Alternate, or Last Events screen as problems or problem information records to DSM/PM.
		Displays the DSM/PM Main menu when you press F5 from the TACL screen.
SF5	DSM/PM	Displays the DSM/PM Create Problem screen and lists as the problem subject any previously marked event on the Primary Events screen. DSM/PM does not submit the problem; it waits for you to take further action.
		Displays the DSM/PM Main menu when SF5 is pressed from the TACL screen.

Table 3-1. Summary of Function-Key Assignments *(page 2 of 3)*

Key	Function	Description
F6	Ack/ Delete	Displays the Events screens, and acknowledges any action or critical events that are on by dimming them. Used on the Network Status Summary screen, deletes selected status items. Used on the Event Configuration screen, deletes that configuration.
F7	Add	Used on the Network Status Summary screen, designates (with the cursor) where to add a new status item.
F8	Freeze	Used on the Events screens, prevents additional events from appearing by freezing the scrolling action. Used on the Network Status Summary screen, stops the regular updates of status information. The word FROZEN is displayed on line 3.
SF8	Thaw	Used on the Network Status Summary and Events screens, cancels any freeze condition applied on the current screen, allowing updates for status or events to resume.
F9	Print	Used on the Events, Last Events, and Network Status Summary screens, prints the information displayed on the current screen. You can enter a destination file name on the option line; otherwise, the printer specified in the Pathway configuration for your terminal is used.
F10	Clip	Used on the Network Status Summary, Events, and Event Detail screens, copies up to 77 characters of text for each selected status item, event message, or line of detail to your clipboard file.
F11	Detail	Used on the Events screens, displays the Event Detail screen with the details for the first event you selected on the screen. Repeated use of F11 displays the other events you selected and returns after the final one.
F12	Update Configuration	Displays one of the four configuration screens (depending on current screen) and displays the current configuration data (see Figure 3-2). On the configuration screens, you use F12 to force acceptance of new configuration data and (in the case of status item configuration) to advance to the next selected item.
F13	Scan	Used on the Status Item Configuration screen, obtains default values for various status items (such as for a new configuration file). Press F12 to cause acceptance of an item (after you have made any desired changes) or SF13 to move on to the next status item.
SF13	Scan Next	Used on the Status Item Configuration screen, displays default values for the next item, using the search parameters you specified when you pressed F13.
F14	Profile	Displays the Profile screen and the names of the current status and event configuration files. On the Profile screen, press F14 to cause acceptance of new (or default) configuration files.

Table 3-1. Summary of Function-Key Assignments *(page 3 of 3)*

Key	Function	Description
SF14	Recover Screen	Used on the Event Configuration, Event Detail, Last Events, Network Status Summary, and Status Item Configuration Events screens, recovers the screen to its last known state.
F15	Help	Displays the Help screen and the first page of help information for the current screen. The Help function key on the TACL screen displays the purpose of each function key.
SF15	Extras	Displays the initial screen of a custom application (if any). Select an event on an Events screen before pressing SF15 to pass event information to the custom application.
F16	Return	Used on the Event Detail screen, returns to the Events screen from which you requested detail.
		Used on the Status Item and Event Configuration screens, returns to the invoking screen without making a configuration change.
		Used on a DSM/PM screen, returns to the ViewPoint screen you were using when you invoked DSM/PM.
SF16	Exit	On any ViewPoint screen, causes the terminal to exit from ViewPoint.
		Used on a DSM/PM screen, returns to the ViewPoint screen you were using when you invoked DSM/PM and logs you off of DSM/PM.

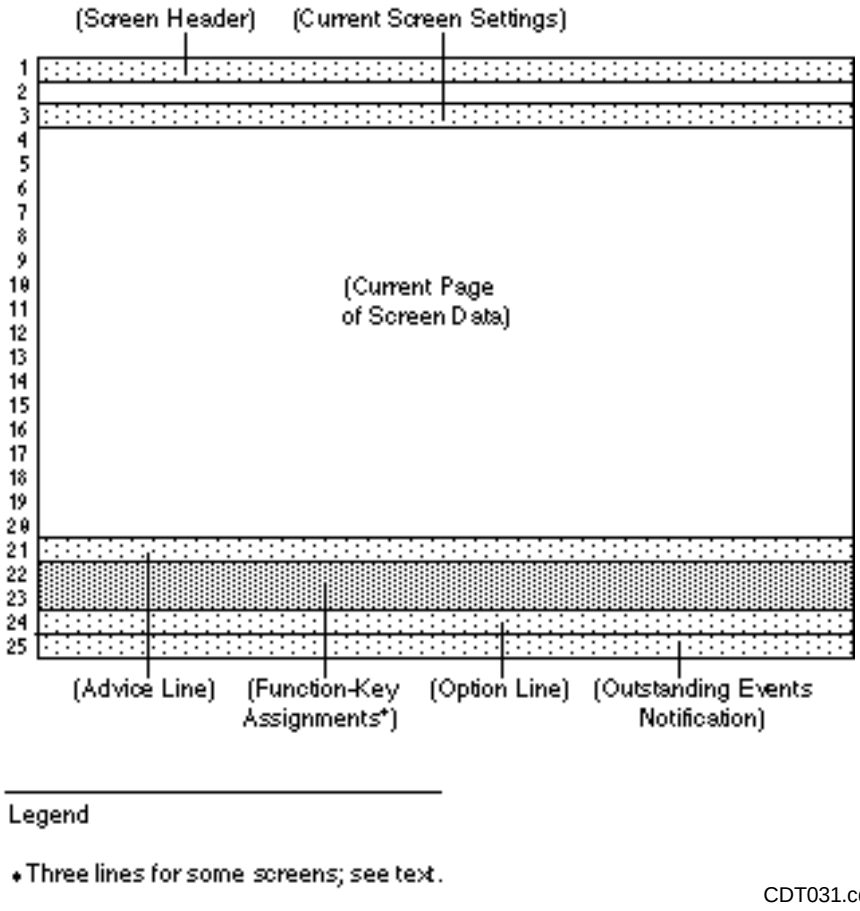
The primary use of the function keys is to display the various screens. In other cases, however, the operation of a key depends on what screen you are in. For specific information in these cases, refer to the screen descriptions that follow later in this section.

Not all the function keys can be used on all screens. A dedicated field of two or three lines near the bottom of the block-mode screens (at the top of the TACL screen) lists all the function keys you can use on the current screen.

Block-Mode Screen Format

Figure 3-1 shows the general format for block-mode screens.

Figure 3-1. Block-Mode Screen Format



Line 1 of a block-mode screen is the screen header. It displays which screen you are using and the time the screen was last updated. Also, for multipage screens, line 1 displays your current page position. Line 2 is blank. Line 3 is blank for some screens; for others, it lists certain settings applicable to the current screen, such as the name of a configuration file currently being used.

For most screens, lines 4 through 20 display the current page of screen data, such as event or configuration information. Line 21 is the **advice line**, which gives you error notices, status messages, or advice messages. Lines 22 and 23 tell you which function keys you can use from this screen and their assignments. For certain other screens (status and event messages), three lines are required to display the function-key assignments—lines 21, 22, and 23. In that case, line 20 is the advice line, and lines 4 through 19 are available to display the current page of screen data (16 lines).

Line 24 is the option line, which is available for supplying information that is meant to be used with the function keys, such as to pass the name of a subject when using the Last Events function key. You can also specify a page number on line 24 to display a particular page of the Primary or Alternate Events screens or the Network Status Summary screen. You can do this from within the screen or from another screen before pressing the function key to select the screen.

Line 25 is special in that it is separately controlled and does not change when you go from screen to screen. This line always displays the total number of outstanding events and a portion of the text of the most recently reported outstanding event. However, the content of the line is cleared whenever you switch between conversational-mode and block-mode screens and is updated only when a function key is used or when screen information is updated. The twenty-fifth line display is not updated for status and events screens that are in the frozen state. A separate field at the end of the line displays what mode you are in—conversational (TTY-FDX) or block (BLOCK).

Screen Navigation

The two main categories of ViewPoint screens are: network presentation screens and configuration screens.

Network Presentation Screens

- Alternate Events screen
- Network Status Summary screen
- Events screen
- Last Events screen
- Primary Events screen
- Event Detail screen

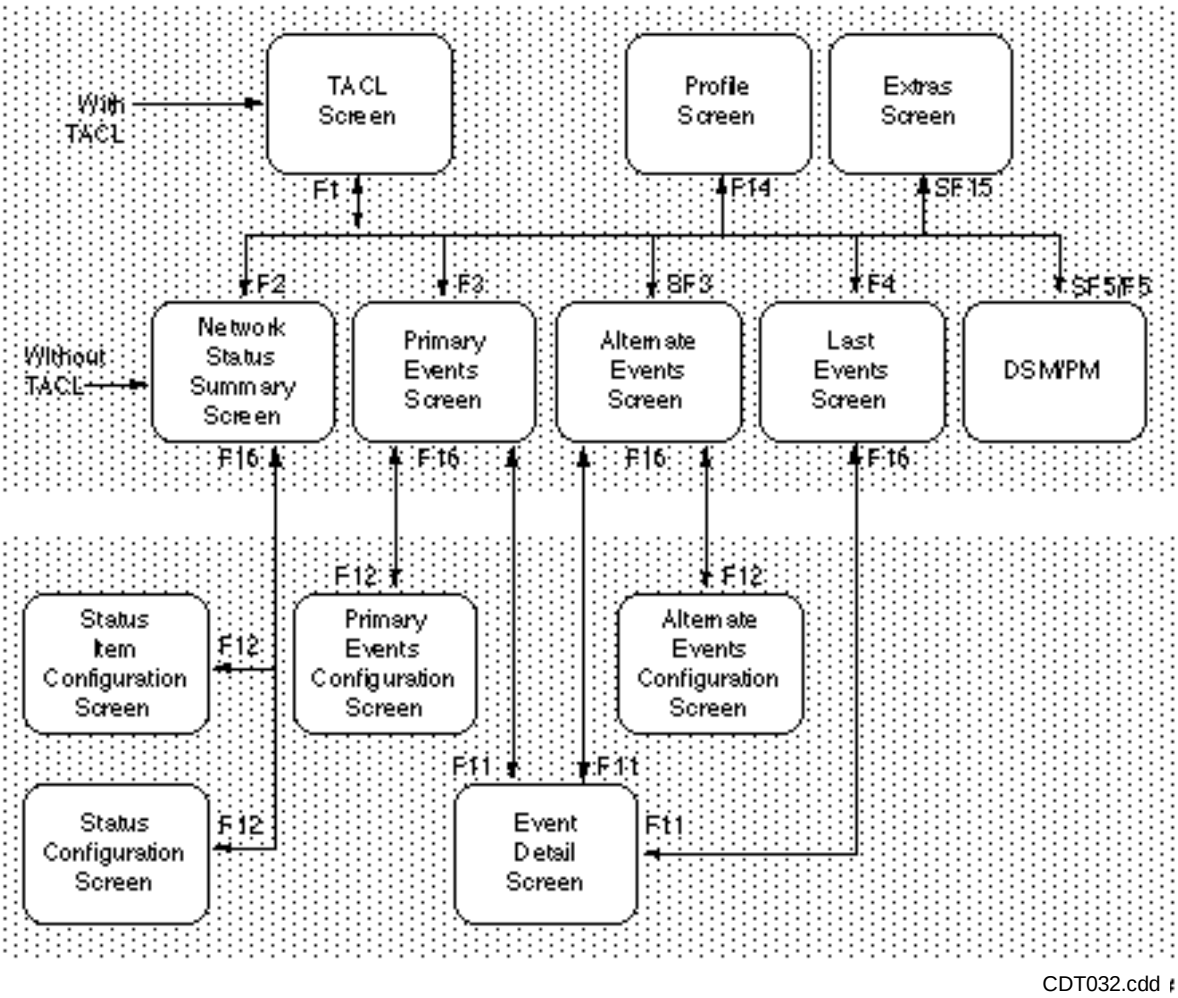
Configuration Screens

- Profile screen
- Primary Events Configuration screen
- Alternate Events Configuration screen
- Status Configuration screen
- Status Item Configuration screen
- TACL screen

Figure 3-2 shows the layout of these screens. Use this figure to help you move among the different screens. When you log on to ViewPoint, if you have access to TACL, you see the TACL screen. From there you can use the function keys shown in this figure to move among the screens shown in the upper half of the figure. The screens are TACL, Profile, Network Status Summary, Primary Events, Alternate Events, Last Events,

Extras. For example, use F3 to go to the Primary Events screen or F14 to go to the Profile screen.

Figure 3-2. Screen Navigation Map



You can also switch to the DSM/PM screens if DSM/PM is configured to run with ViewPoint. For an example of how to access DSM/PM from ViewPoint, see Section 2, Using ViewPoint.

If you do not have access to TACL, you can move among the remaining six screens in the upper half of the diagram (including the Profile and Extra screens). Access the screens by using the function keys.

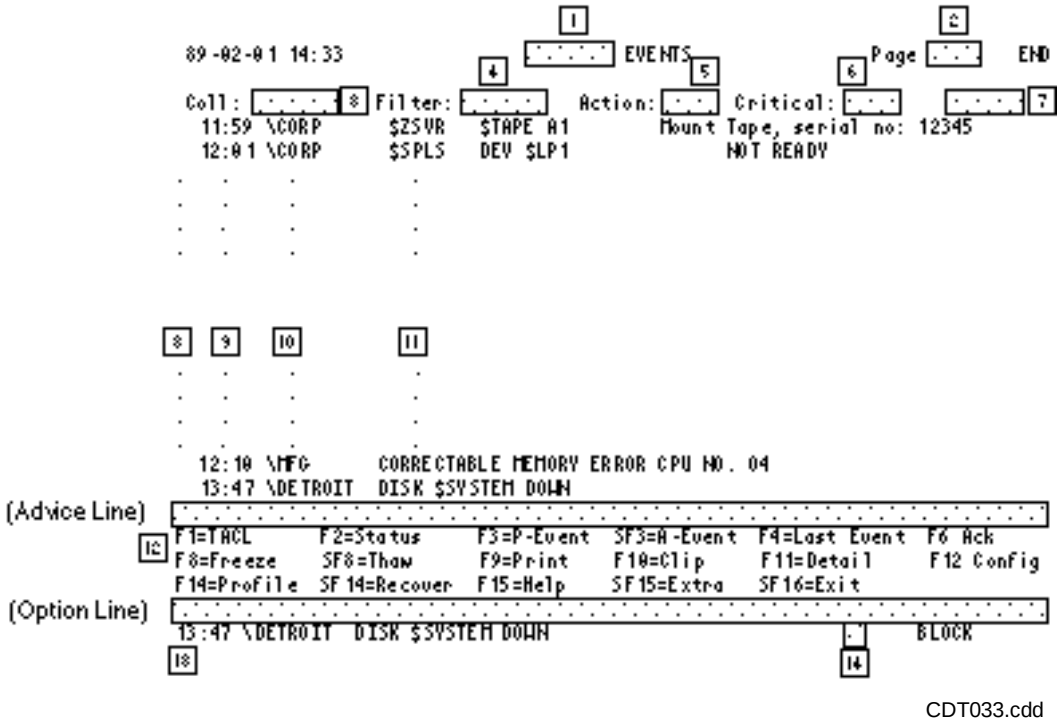
You can access the Event Detail screen and the four configuration screens shown in the lower half of the Figure 3-2 from only the four main status and events screens. Note that you always use F12 to access any of the four configuration screens—the screen you move to depends on what screen you are in and what you enter on that screen.

The remainder of this section describes nine of the 12 ViewPoint screens. They are described in alphabetical order. Only nine are shown because the Primary Events screen and the Alternate Events screen have similar format, so they are listed as a single Events screen; the Primary Event Configuration screen and the Alternate Event Configuration screen are listed as a single Event Configuration screen; and the last screen, called Extra, is only displayed if you have a customized ViewPoint.

Events Screen

Figure 3-3 shows the format for the Events screen. This format is the same for the Primary Events screen and the Alternate Events screen.

Figure 3-3. Events Screen Format



CDT033.cdd

Field Descriptions

1. **Display**. Name of the event display currently selected. Press F3 to select the primary events screen or SF3 for the alternate events screen. This field displays either the word PRIMARY or the word ALTERNATE, depending on which screen you selected.

2. **Page**. Page number of event messages you are currently viewing. The number of the END page (the only one that is updated as new event messages come in) continues to increment until it reaches the configured size of the cache; then, it remains at that number. A field following the page field tells you that you are on the END page. Earlier pages are numbered downward from the END page. Search through the various pages of earlier events. Note that this is the end page, as indicated in the upper-right corner of this screen.

   - To observe earlier events, press PREV PAGE (or PG UP on some terminals).
   - When the current page is not the last page, press NEXT PAGE (or PG DN on some terminals) to observe later events.
   - To select the first page, press SHIFT-PREV PAGE (or ALT-PG UP). Alternatively, you can type the page number in the option line and then press F3.
   - To select the END page, press SHIFT-NEXT PAGE or by selecting a page number of 20 (or greater). Notice that only the END page gets updated as new event messages arrive.

3. **Collector**. Name of the event collector process that collected the event messages you are viewing.

4. **Filter**. Name of the filter file that was used to pass the event messages you are viewing. Only the file-name portion of the name is shown, excluding the volume and subvolume portions.

5. **Action**. Number of unique outstanding action events for the current display (primary or alternate). If action events are not configured to be displayed, the number in this field is dimmed on the screen.

   Note that two or more action events with the same subject, the same ACTION-ID token value, and from the same system, are counted as a single unique action event. When more than one of these duplicate events are displayed, only the most recent is highlighted.

6. **Critical**. Number of outstanding critical events for the current display (primary or alternate). If the event display configuration did not include critical events, the number in this field is dimmed on the screen.

7. **Frozen**. This field is normally blank. It displays the word FROZEN after you press F8 (Freeze screen) while viewing the END page. (Using F8 is meaningful only on the END page because only that page scrolls as new messages arrive.) While the display is frozen, new messages accumulate in the event cache. The messages are not displayed until you press SF8, Thaw screen. This is also true for updates to the twenty-fifth line.

8. **Event Selection**. This is a one-character field (per line) that allows you to select individual event messages for copying to your clipboard file, for deleting outstanding events, or for requesting details about an event.

Select any single event message by positioning the cursor in this field, or enter any nonblank character in this field to select any number of events, and press the appropriate function key for the operation you want.

To avoid possible inaccurate marking due to sudden scrolling as new messages arrive, you should first freeze the screen using F8.

9. **Generation Time**. Time of day when the event displayed on that line was generated.
10. **System**. Network node that originated the message.
11. **Message**. Text of the event message (or the first 62 characters). Outstanding action event messages are highlighted with inverse video display, and outstanding critical event messages are highlighted with full brightness. All other messages are displayed with half brightness. For longer messages, display the remainder of the text by selecting that line (cursor or nonblank character in column 1) and pressing F11, Event Detail screen.
12. **Function keys**:

F1	TACL
F2	Status
F3	P-Event
SF3	A Event
F4	Last Event
F5*	DSM/PM
SF5*	DSM/PM
F6	Ack
F8	Freeze
SF8	Thaw
F9	Print
F10	Clip
F11	Detail
F12	Config
F14	Profile
SF14	Recover
F15	Help
SF15	Extra
SF16	Exit

∗This function key is active only if you have configured ViewPoint to run with DSM/PM; even when it is active, however, you will not see it in the list of functions keys appearing at the bottom of your screen. For more information on using this function key, see its listing under “Function Key Notes” in this section.

When you press a function key to go to another screen, and return to the current Events screen, your screen context is preserved; that is, the current page number, any marked event messages, and the frozen or thawed state are unchanged when you return to the Events screen.

Function Key Notes

Table 3-2 describes the operation of some of the function keys and special requirements for using them.

Table 3-2. Function Key Notes (Events Screen) *(page 1 of 2)*

Function Key	Special Requirements
F4	Before displaying the Last Events screen, you must select a subject. Select a subject by positioning the cursor (or entering a nonblank character) in the selection field preceding a message that identifies the subject you want to know about. Then, press F4 to request the Last Events screen. Select a message whose subject token is the subject for which you want to see recent events. For an explanation of subject tokens, see the description of the Last Events screen.

Table 3-2. Function Key Notes (Events Screen) *(page 2 of 2)*

Function Key	Special Requirements
F5	Used in the following two ways: ● Displays a particular DSM/PM screen. Type the name of the screen (or any valid alias) on the option line, and press F5. ● Submits events as problems. Mark the events by typing any character next to them, and press F5. For more information on submitting events as problems, see Section 2, "Submitting Events as Problems or Problem Information Records."
SF5	Used in the following two ways: ● Sends an event marked on a ViewPoint event screen to DSM/PM as a created problem requiring additional information. Mark the event by typing any character next to it, and press SF5. The object name associated with the event you mark is displayed as the problem subject on the screen. DSM/PM does not automatically submit the problem; it waits for you to take further action. For more information on adding a marked event to DSM/PM see, Section 2, "Adding an Event Without Submitting It." ● Sends multiple events marked on a ViewPoint screen to DSM/PM as problem information records. Mark the events by typing any character next to them, and press SF5. The DSM/PM Create Problem screen displays the marked events appearing as problem information records at the bottom of the screen. You can create a problem using all the functions available on the screen. For more information on submitting marked events as problem information records, see Section 2, "Submitting Events as Problems or Problem Information Records."
F6	Acknowledges a critical or action event. The event message is dimmed to indicate that it has been acknowledged. The message remains on the screen if the view of critical and action events includes acknowledged events. The highlighting is eliminated, and the message appears in dimmed (half-bright) video display or in grey on a color display. Also, the event is no longer considered outstanding for the line-25 count. An action event is normally acknowledged by a completion message that is triggered when you physically perform the requested action. The message is acknowledged at all operator terminals. However, when you use F6, the results are different depending on which screen you are on when you press F6. If used on the Primary Events screen, the event is acknowledged at all operator terminals; if on the Alternate Events screen, it is acknowledged only at your terminal.

Table 3-3 summarizes the acknowledgment process for critical and action events. If an event is dimmed everywhere (that is, on all operator terminals running on this instance of the ViewPoint application), it is not automatically dimmed on Alternate Events screens that are opened after the dimming.

Table 3-3. Summary of Event Acknowledgment for Critical and Action Events

Event	Dimmed Locally	Dimmed Everywhere
CRITICAL EVENTS		
Acknowledged on Primary Events Screen		X
Acknowledged on Alternate Events Screen	X	
ACTION EVENTS		
Acknowledged Automatically		X
Acknowledged Manually		
On Primary Events Screen		X
On Alternate Events Screen	X	

Action events are also dimmed when the requested action has been taken; that is, when the action is completed. By default, when an action is completed, ViewPoint also displays an action-completion message. This message might appear many pages later (after the original action event-message). In some cases, the action-completion message is suppressed by a SUPPRESS^DISPLAY token in the action-completion event sent to the ViewPoint filter. (Refer to the subsection, "Basic Default Filtering Rules" in Section 6, Customizing ViewPoint for a discussion of how the default filters handle suppressed messages.)

F12 displays the configuration screen for whichever display you are currently viewing, either primary or alternate. Thus, if you are viewing primary events, and you want to reconfigure the alternate events display, you must first use SF3 to access the Alternate Events screen. From there, you can display the desired configuration screen with F12.

1. **Last Outstanding Event**. Most recently received critical or action event (or a portion of the event). Figure 3-3 shows a critical message just received and displayed both as the last message from the cache and as the most recent outstanding event on line 25. Note that this same message appears on all other screens in line 25.

2. **Outstanding Count**. Total number of outstanding critical or action events in the primary cache. Outstanding events that roll off the end of the cache decrease this count, and the line-25 message displays "Outstanding event unanswered."

   Note that duplicate action events (action events with the same subject, the same ACTION-ID token value, and from the same system) are counted as a single unique action event. As a result, you might see more action events displayed on the screen than are counted in this field.

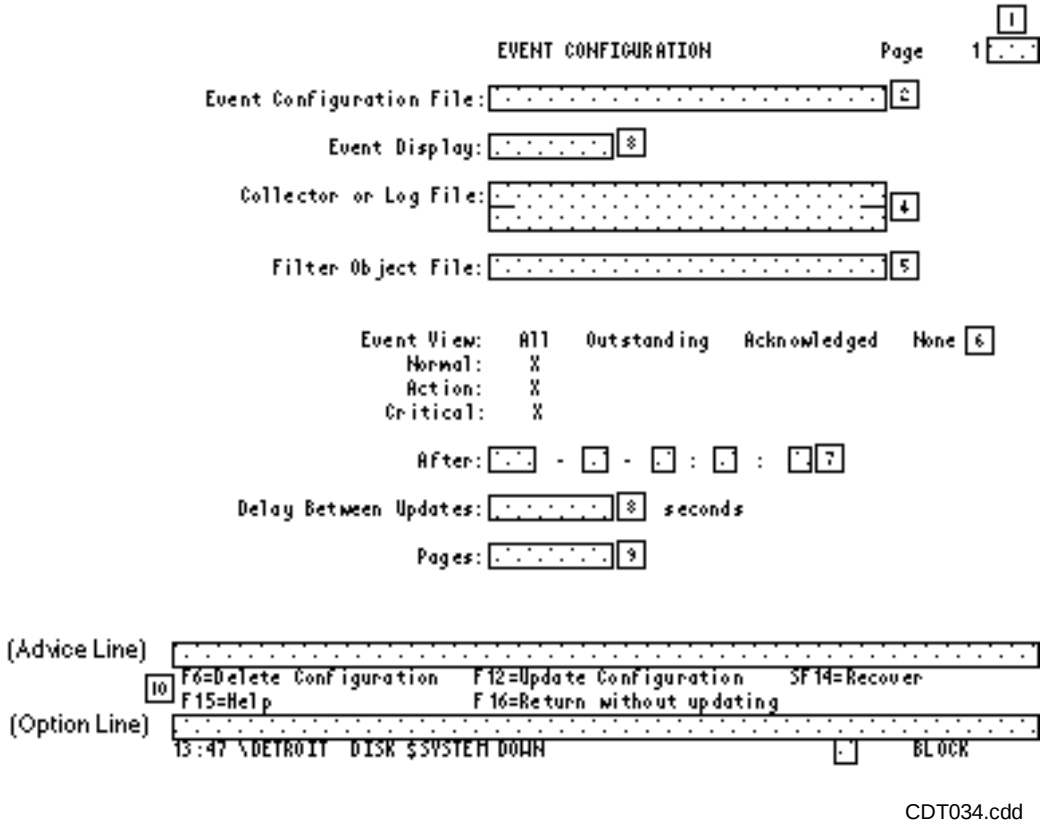
Event Configuration Screen

This screen can have one or two pages, depending on whether it was requested from the Primary Events screen or from the Alternate Events screen. The Alternate Event Configuration screen has a second page that is used to pass (SPECIFY) parameters for filtering events displayed on the Alternate Events screen.

Page 1 of the Event Configuration Screen

Figure 3-4 shows the format for the first page (Page 1) of the Event Configuration screen. Figure 3-5 (later in this subsection) shows the format for the second page.

Figure 3-4. Event Configuration Screen, Page 1 Format



CDT034.cdd

Field Descriptions

1. **Page 1**. Page identifier that tells you whether this is the END page or whether MORE pages are available. It specifies MORE only if this is an Alternate Event Configuration screen. You can use the NEXT PAGE (or PG DN) key to go to the second page (if there is one) and the PREV PAGE (or PG UP) key to return to the first page.
2. **Event Configuration File**. Name of the configuration file you are currently modifying. If you want to select a different one, you need to use the Profile screen.

3. **Event Display**. This field, which is not modifiable, tells you whether this configuration applies to the primary display (PRIMARY) or to the alternate display (ALTERNATE).

4. **Collector or Log File**. Name of the collector or log file used. This field is modifiable for configuring only an Alternate Events screen. (For the Primary Events screen, the collector name or names are preconfigured; you cannot change them dynamically. However, they can be modified in the ViewPoint configuration, as described in Section 7, Installation, Configuration, and Startup.)

   You can specify up to five collectors (or one log file) in this field. If event collection is from the primary collector in your local system, type $0 in this field. To specify event collection from any other event collectors (or a log file), enter the name (or names) in this field. Each name must be preceded by the name of the system where that collector (or the log file) resides.

   Enter multiple collector names as a list; the names must be separated by commas or spaces. When remote collectors are named, ViewPoint retrieves events directly from the collectors on the remote systems. When there are multiple collectors, the EMS consumer distributor merges the events from the collectors rather than using EMS forwarding distributors to forward the events to the local system.

   Note that transferring unfiltered events from multiple collectors on remote systems is less efficient than using forwarding distributors to forward filtered events to your local system; such unfiltered transfers could significantly increase network traffic and degrade response time on the network.

5. **Filter Object File**. Filters restrict events that are passed from the event collection and distribution mechanism into your event cache. The filter used for the primary display is not replaceable. You can, however, specify a filter for your alternate display. Enter in this field the file name of the filter you want to use. To qualify the file name, use the volume and subvolume names. Refer to the *EMS Manual* for information about filters and how to create them. To pass information to the specified filter, use page 2 of this screen (see Figure 3-5).

   Note. If you write a custom filter, include the default code for parameters to prevent problems if a noncustom parameter is specified. You can copy this default code from Figure 6-2 in Section 6, Customizing ViewPoint.

6. **Event View**. This field allows you to select any combination of event types (normal, action, and critical) to be displayed on the screen. Normally, ViewPoint is set to display all types of events.

   For each event type, you can specify that all of them are or none of them is displayed. In the case of action and critical events, you can specify that only outstanding or acknowledged events are displayed.

   You select the combination of events you want to display by inserting any character under one of the appropriate options for each event type: All, Outstanding, Acknowledged, or None. You can only mark one option for each event type.

You must return to this screen each time you want to change your event view. Your most recent selection remains in the configuration file until the next time you change it.

7. **After**. These fields are modifiable only for alternate-event configurations, since the primary-event display is for current events only. You use these fields to specify a starting time (or date and time, or just date) to view events in historical mode; in fact, your use of this field is one indication to ViewPoint that you want historical mode. (Another one is your specifying a log file.) The status line on the Alternate Event screen keeps you informed of ViewPoint progress by displaying historical events on the screen.

   Enter the desired date and time in the following format:

```
yyyy-mm-dd.hh:mm:ss
```

yyyy

The year in four numeric values; for example: 1990.

mm

The month of the year; for example: April is 04 or 4.

dd

The day of the month; for example: 15.

hh

The hour according to a 24-hour clock; for example: 13 is 1 p.m.

mm

The minutes; for example: 30.

ss

The seconds; for example: 05.

If the fields are left blank, they default to the following:

time—the current time
date—the current date

For date, any blank portions are filled in with the current date, and the year can be specified with two or four digits, such as 93 or 1993.

1. **Delay Between Updates**. You can use this field to specify a minimum interval (in seconds) between screen updates. The default interval is 10 seconds.

2. **Pages**. You can use this field to increase or decrease the size of the alternate-events cache. (The content of field 2 tells you which cache you are modifying.) The

default cache size is 20 pages. (The pages shown in this section refer to 16-line event-screen pages; thus the default cache size holds 320 event messages.)

3. **Function keys**

F6	Delete configuration
F12	Update configuration
SF14	Recover
F15	Help
F16	Return without updating

Function Key Notes

The following notes describe the operation of some of the function keys and special requirements for using them.

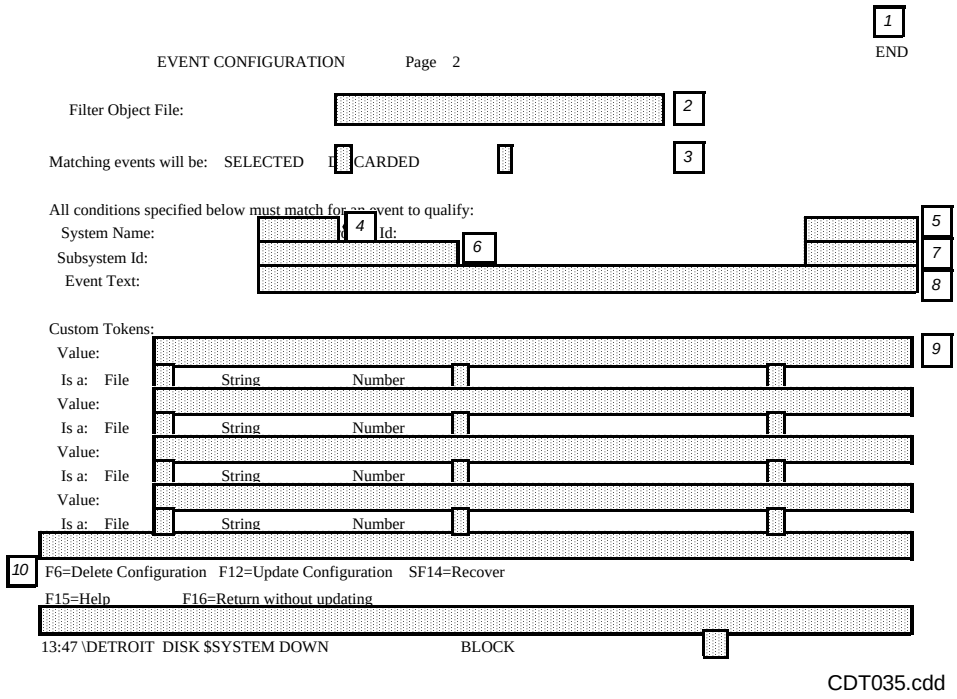
Table 3-4. Function Key Notes (Event Configuration Screen)

Function Key	Special Requirements
F12	To make any changes made to this screen effective, press F12. ViewPoint makes the changes in the configuration file and returns you to the Events screen.
F16	If you use F16 to return to the Events screen, any changes you made on this screen are ignored and lost.

Page 2 of the Event Configuration Screen

This screen allows you to pass information to the current ViewPoint filter that selects the alternate events for display. You use this screen to dynamically configure the current filter by passing it parameters to use in determining what events to pass to ViewPoint for display on the screen. (Refer to Section 6, Customizing ViewPoint for a description of how the default filter, FLTRALT, or any other custom filter uses the information from this screen.)

Figure 3-5. Event Configuration Screen, Page 2 Format



Field Descriptions

1. **Page 2**. This field, which is not modifiable, tells you that this is page 2, the END page. This page is displayed only for alternate event configuration.

2. **Filter Object File**. This field, which is not modifiable, tells you the name of the filter object file currently in effect. Usually, this is the default filter, FLTRALT, provided with ViewPoint. Select a different filter object file on page 1 of the Alternate Event Configuration screen.

3. **Matching Events**. This field indicates whether you want ViewPoint to select or discard events that match subsequent values entered on the screen. You can mark either SELECTED or DISCARDED, but not both. If you mark SELECTED, the filter specified at the top of the screen returns to ViewPoint only those events that match subsequent values. If you mark DISCARDED, the filter returns all events except those that match values specified on this screen. If you mark neither, the current filter determines what events are returned; the default filter, FLTRALT, returns all events.

4. **System Name**. This field can contain a system name, which must start with a backslash (\). The system name of the process that reports an event must match the specified name in order for that event to be selected or discarded by the filter. The system name you enter can be in uppercase or lowercase letters.

5. **CPU or Process ID**. This field can contain a process ID or a CPU number. The ID or CPU number of the process that reports an event must match the specified ID or CPU number to qualify that event to be selected or discarded by the filter. You specify a process ID as a process name (1 to 5 alphanumeric characters preceded by $) or as two integers (*cpu*-the CPU number, and *pin*-the process ID number). If you specify only one number, ViewPoint interprets the number as the CPU number and compares it to the CPU number of the reporting process.

6. **Subsystem ID**. This field can contain a subsystem ID. The subsystem ID that reports an event must match the specified ID to qualify that event to be either selected or discarded by the filter. The comparison is not case-sensitive; that is, uppercase and lowercase letters are searched as the same character. You specify the subsystem ID in external format without version information. The format is:

   owner.ss

   owner is 1 to 8 alphanumeric characters or hyphens, and the first character must be a letter.

   ss is either the subsystem number or the subsystem name. A subsystem number is a 16-bit signed integer; a subsystem name is 1 to 8 alphanumeric characters or hyphens, the first of which must be a letter. Examples are:

   ```
   TANDEM.PATHWAY
   TANDEM.52
   ```

   Note that the subsystem ID you specify in this field does not contain version information; this differs from the subsystem ID in an SPI specification.

7. **Event Number**. This field can contain an event number. The event number of an event must match the specified event number for that event to qualify to be selected or discarded by the filter. You specify the event number as a signed or unsigned 16-bit integer of the form:

   [+ / -] [%] *number*

   You can use the % prefix to specify an octal base. The value of *number* must be in the decimal range -32768 to 32767.

8. **Event Text**. This field can contain event text. The text of an event must match the specified text in order for the event to be selected or discarded by the filter. ViewPoint compares the entered text to the text generated by EMSTEXT (less header text) for the event. Enter the event text as any alphanumeric characters, with or without enclosing quotation marks. If you do enclose the text within quotation marks and the value contains quotation marks, you must double the contained quotation marks. Trailing blanks are discarded unless the text containing the blanks is enclosed within quotation marks. You can use an asterisk (*) to match zero or more alphanumeric characters, and you can use a question mark (?) to match any one character. The event text can be in uppercase or lowercase letters.

9. **Custom Tokens**. These fields contain values and types for up to four custom parameters that can be passed to the current filter. The default filter, FLTRALT, does not use these parameters unless logic has been added to the filter to test for the custom parameters. If the logic to test these parameters is in a filter other than FLTRALT, then that filter must be named on page 1 of the Event Configuration

screen for these parameters to be effective. (Refer to Section 6, Customizing ViewPoint for a discussion of how to add logic to FLTRALT or how to write a custom filter.)

You specify custom parameters as a value followed by a type. For each parameter value, you must indicate its type. The type you mark determines how the value is interpreted. Table 3-5 describes the value for each type.

Table 3-5. Value and Type for Custom Parameters

Type	Value
File	You must specify the value as a file name.
String	You can specify the value as any string, with or without enclosing quotation marks. If you enclose the value in quotation marks and the value contains quotation marks, you must double the contained quotation marks. Trailing blanks are discarded unless the value, including the blanks, is enclosed in quotation marks.
Number	You can specify the value as a 64-bit signed or unsigned integer of the form: [+ / -] [*prefix*] *number* You can use any of the following *prefix* values to override the default numerical base: % = Octal # = Decimal %h or %H = Hexadecimal %b or %B = Binary The number must not contain embedded commas; and the value of *number* must be in the decimal range –2147483648 to 2147483647.

10. **Function keys**

F6 Delete configuration

F12 Update configuration

SF14 Recover screen

F15 Help screen

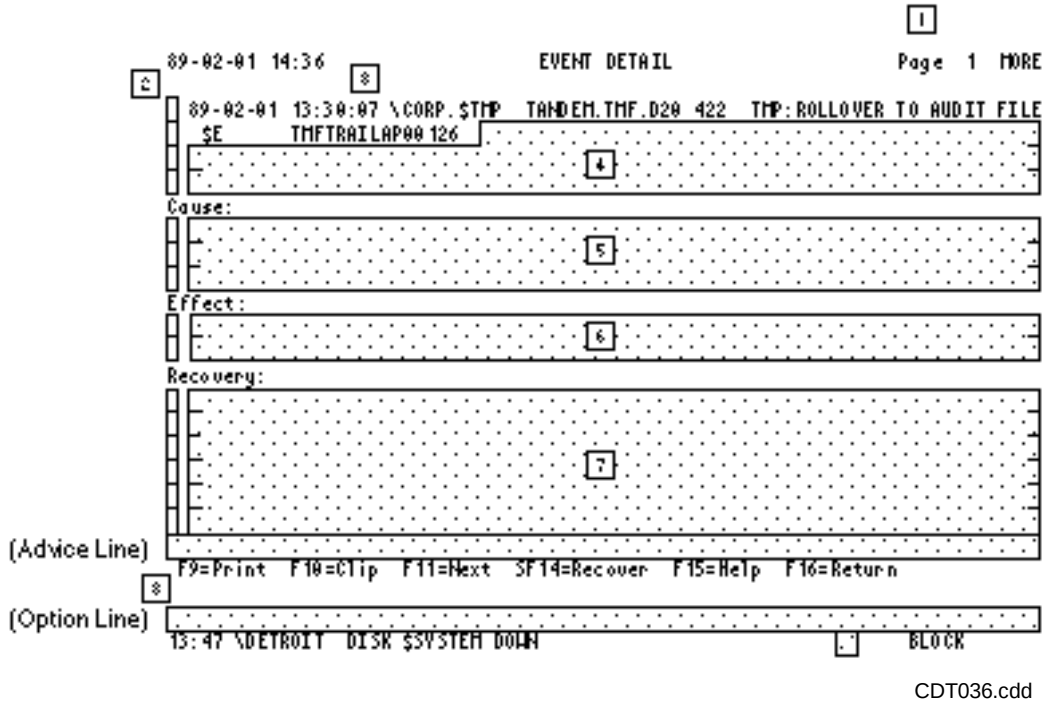
F16 Return without updating

Note. If you use F16 to return to an events screen, any changes you have made on this screen are ignored and lost. To make the changes effective, and return to the events screen, press F12, Update configuration.

Event Detail Screen

Figure 3-6 shows the format for the Event Detail screen.

Figure 3-6. Event Detail Screen Format



CDT036.cdd

Field Descriptions

1. **Page**. Page you are currently viewing. This is followed by the word MORE if there are more pages or by the word END if you are viewing the final page.

   You can scroll forward with the NEXT PAGE (or PG DN) key and backward with the PREV PAGE (or PG UP) key. SHIFT-NEXT PAGE (or ALT-PG DN) takes you directly to the END page and SHIFT-PREV PAGE (or ALT-PG UP) returns you to the first page.

2. **Copy Selection**. Enter a nonblank character in this field in front of those lines you want copied to your clipboard file. Then press F10 to have the lines copied.

3. **Origin Header**. This field gives the following information about the origin of the event message: date, time, process name or identification, subsystem identification, and event number.

4. **Event Text**. This field gives a complete version of the event message.

5. **Cause**. This field displays a probable cause (if any) of the event.

6. **Effect**. This field displays any possible effect on the system that this event might have caused.
7. **Recovery**. This field displays the recommended action (if any) for you to take for this event.
8. **Function keys** :

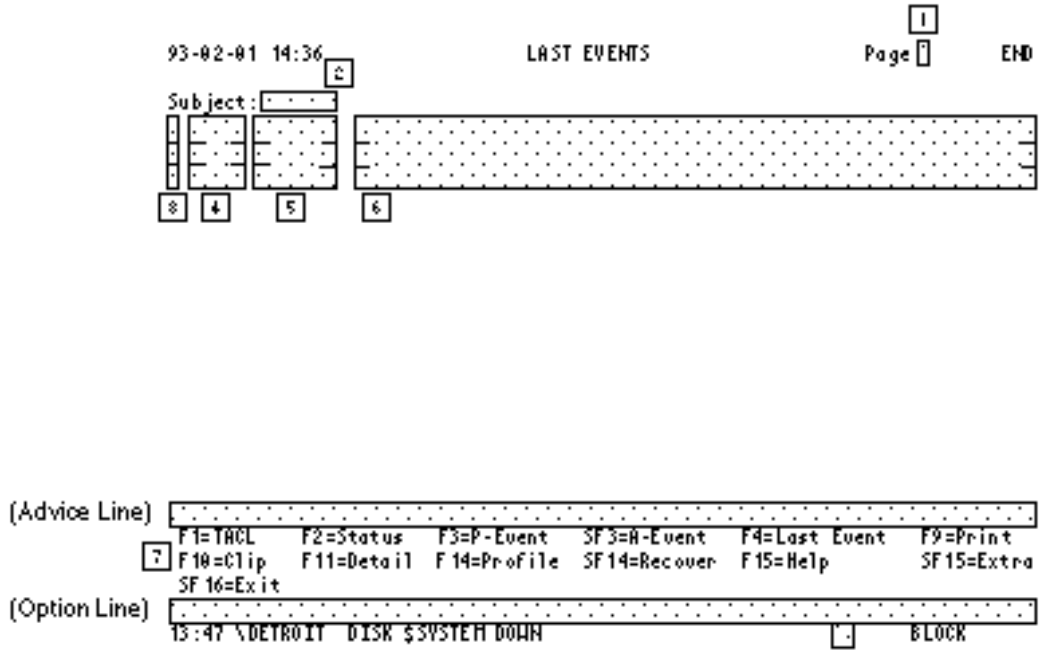
F9	Print screen
F10	Clip
F11	Next
SF14	Recover
F15	Help
F16	Return

Note. If you have selected several events on the previous screen for detail viewing, press F11 to step from one to the next. After you view the final selected event, press F11 to return to the previous screen. If you want to return immediately, use F16.

Last Events Screen

Figure 3-7 shows the format for the Last Events screen. Field descriptions and function-key notes follow.

Figure 3-7. Last Events Screen Format



CDT037.cdd

Field Descriptions

1. **Page**. Page number of the last-events display you are currently viewing. A field following the page field tells you that you are on the END page. If there are more pages to be viewed, the initially viewed screen (with the most recent events) has the highest page number, and earlier pages are numbered downward.

   Earlier pages are numbered downward. Search through the various pages of earlier events. Note that this is the end page, as indicated in the upper-right corner of this screen.

   - To observe earlier events, press PREV PAGE (or PG UP on some terminals).
   - When the current page is not the last page, press NEXT PAGE (or PG DN on some terminals) to observe later events.
   - To select the first page, press SHIFT-PREV PAGE (or ALT-PG UP). Alternatively, you can type the page number in the option line and then press F3.
   - To select the END page, press SHIFT-NEXT PAGE or by selecting a page number of 20 (or greater). Notice that only the END page gets updated as new event messages arrive. The field following the Page field reads MORE when the END page is not displayed.

2. **Subject**. Name of the subject associated with these event messages. This is the name you selected on the screen either by entering it on the option line or by marking an event message with the cursor or a nonblank character. To change the subject, type a new subject name on the option line and press F4.

   The subject name you select refers to the subject token associated with that event. (For an explanation of subject tokens, refer to the *EMS Manual*.) Each event can have several different subject tokens associated with it. ViewPoint builds a database of events based on particular subject tokens. Then, when you specify a particular subject, ViewPoint looks in this database and displays all the event messages that are listed for this subject.

   To build each events database, ViewPoint scans each event to determine if it has one of the following valid types of subject tokens associated with it: character, crtpid, device, filename, fname32, unsigned integer, or transid. If none of these types of subject tokens is found, the event is not entered in a database. For each subject token that is found, the event is entered in the database of events for that subject token.

   ViewPoint follows these guidelines in determining suitable subject tokens and how they are formatted on the screen:

   - If the subject token is a string of characters and there is a manager token (a filename token), the subject token is formatted as follows:

     ```
     \NODE.$MANAGERNAME.STRING
     ```

   - ViewPoint converts the subject tokens of types crtpid, device, and filename to external form using the procedure FNAMECOLLAPSE. ViewPoint converts the subject token fname32 to external form using the procedure FNAME32COLLAPSE. ViewPoint formats any subject token of type filename that is not qualified with a node name into the following format:

     ```
     \NODE.$VOL.SUBVOL.FILENAME
     ```

     If node names are not available, ViewPoint uses the three-digit system number for the node. If the system is local and not part of a network, the system name does not appear in the prefix.

   - An unsigned integer subject token is valid only if its token code is ZEMS^TKN^LDEV or ZEMS^TKN^SYSPID. An ldev code is converted to a device name using GETDEVNAME. If no device name is available, a device number is used in the format

     ```
     \NODE.$LDEV
     ```

     A syspid code is converted to external form using FNAMECOLLAPSE and is formatted as follows:

     ```
     \NODE.CPU,PIN
     ```

     The syspid token is present in SENDOPMSG events.

   - A transid subject is converted to external form using TRANSIDTOTEXT.

- Subjects appear in uppercase letters so that they can be searched regardless of their capitalization.
- For each suitable subject found, ViewPoint makes an entry in the last events database using the subject as a key. Each subject included in the database has a maximum of 80 characters; longer subjects are truncated to 80 characters.

3. **Event Selection**. Enter a nonblank character in this field to select one or more event messages that you can copy to your clipboard file by pressing F10 or invoke detail by pressing F11.
4. **Log Time**. Time the event on that line was generated.
5. **System**. Network node that originated the message.
6. **Event Message**. Text of the event message. Displays up to 62 characters. If the event message is longer than 62 characters, you can examine the remainder of the text by positioning the cursor in column 1 of the desired line, and pressing F11. This displays the Event Detail screen where you can see the entire message.
7. **Function keys**

F1	TACL
F2	Status
F3	P-Event
SF3	A-Event
F4	Last Event
sup2(*)F5	DSM/PM
sup2(*)SF5	DSM/PM
F9	Print
F10	Clip
F11	Detail
F14	Profile
SF14	Recover
F15	Help
SF15	Extra
SF16	Exit

sup2(*)This function key is active only if you have configured ViewPoint to run with DSM/PM; even when it is active, however, you do not see it in the list of function keys appearing at the bottom of your screen. For more information on using this function key, see its description under “Function Key Notes,” in the following table.

Function Key Notes

Table 3-6 describes the operation of some of the function keys and special requirements for using them.

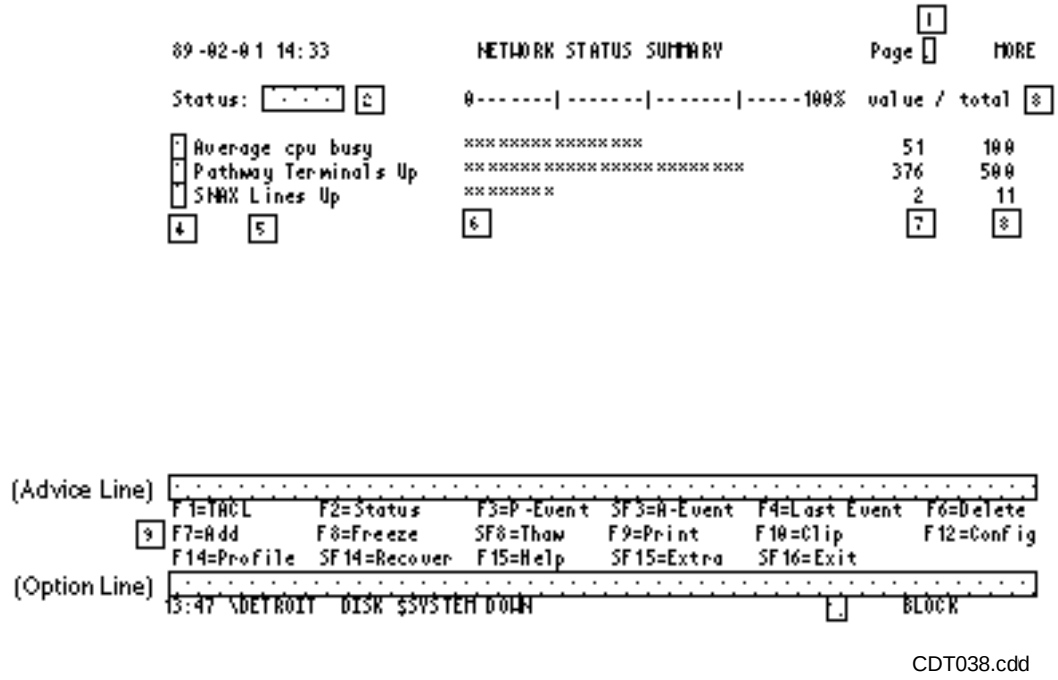
Table 3-6. Function Key Notes (Last Events Screen)

Function Key	Special Requirements
F4	Before displaying the Last Events screen, you must select a subject. Select a subject in one of two ways: (1) by positioning the cursor (or entering a nonblank character) in the selection field preceding a message that identifies the subject for which you want to perform a search. Then, press F4 to request the Last Events screen. Select a message whose subject token is the subject for which you want to see recent events. See the description of the Subject field in the previous subsection, "Last Events Screen," for an explanation of subject tokens; (2) by entering the subject name (if you know it) it directly on the option line (line 24). Then press F4 to request the Last Events screen. Subject names must be less than 78 bytes. Some events have a blank subject name—a null subject. You can display all events with a null subject by positioning the cursor at the option line, which you leave blank, and pressing F4 to request the Last Events screen. NOTE: The event messages displayed for the selected subject do not get updated automatically as new messages are logged for that subject. When you want an update for the subject, press F4 again.
F5	Has two different uses: ● Displays a particular DSM/PM screen if you type the name of the screen (or any valid alias for it) on the option line, and press F5. ● Submits an event as a problem if you mark the event by typing any character next to the event, and press F5. For more information on submitting events as problems, see Submitting Events as Problems or as Problem Information Records in section 2.
SF5	Has two different uses: ● Adds a marked event as a problem to DSM/PM without submitting it, if you mark the event by typing any character next to the event, and press SF5. The object name associated with the event you marked will be displayed as the problem subject on the screen. DSM/PM does not automatically submit the problem, but is waiting for you to take further action. For more information on adding a marked event to DSM/PM, see Adding an Event Without Submitting It, in section 2. ● Adds and submits a marked event as a problem information record to DSM/PM, if you mark the event by typing any character next to it, and press SF5. The DSM/PM Create Problem screen displays with the marked event showing as a problem information record at the bottom of the screen. You can create a problem using all the functions available on the screen. For more information on submitting a marked event as a problem information record, see Submitting Events as Problems or as Problem Information Records in section 2.

Network Status Summary Screen

Figure 3-8 shows the format for the Network Status Summary screen.

Figure 3-8. Network Status Summary Screen Format



CDT038.cdd

Field Descriptions

1. **Page**. Page number you are currently viewing. This is followed by the word MORE if there are more pages or by the word END if you are viewing the final page.

   You can scroll forward with the NEXT PAGE (or PG DN) key and backward with the PREV PAGE (or PG UP) key. SHIFT-NEXT PAGE (or ALT-PG DN) takes you directly to the END page and SHIFT-PREV PAGE (or ALT-PG UP) returns you to the first page. You can go directly to a particular page by entering the page number on the option line and pressing F2.

2. **Status**. Name of the status configuration file you are currently using.

3. **Frozen**. Normally, this field displays the headings for the value and total fields (see Steps 7 and 8 below); however, when you press F8, the word FROZEN blinks in this field. While the screen is in the frozen state, no status updates and no twenty-fifth-line event updates can occur. When you press SF8, Thaw screen, status updates and twenty-fifth-line event updates resume and the FROZEN message disappears.

4. **Item Selection**. Selection of displayed status items for deleting, adding, copying, and configuring. To select more than one status item, type any nonblank character in this field in front of each status item you want to select. Or, to select only one item, position the cursor in this field in front of the status item you want, then use the appropriate function key to operate on the selected items. (See function-key descriptions for F6, F7, F10, and F12 later in this subsection.)

5. **Item Description**. Description of each status item as entered in the Item Description field of the Status Item Configuration screen.

6. **Percentage Graph**. Graphical representation (using asterisks) of the current status measurement for each status item as a percentage—either as a total (such as the number of units up or down) or as a busy value (the unit is x% busy over a measured period of time). Each group of eight asterisks represents 25% (or 3.125% per asterisk).

7. **Value**. Exact measured value for each status item. It represents either a count of units or a percentage busy value, depending on the particular kind of status item. If threshold values have been set for a given item (see the description of the Status Item Configuration screen later in this section) and if the value falls outside those values, the display highlights the entire line in inverse video (or in color if you are using a color monitor).

8. **Total**. Basis currently being used for computing the percentage values on the graph. Typically, this value is the maximum configured number of units (for items that are based on unit counts), or 100 for items that give percentage busy values. However, the basis might have been deliberately set to some lesser value to provide better resolution in cases where measured percentages are expected to be very low at all times. (See the description of the Status Item Configuration screen later in this section.)

9. **Function keys**

F1	TACL
F2	Status
F3	P-Event
SF3	A-Event
F4	Last Event
F6	Delete
F7	Add
F8	Freeze
SF8	Thaw
F9	Print
F10	Clip
F12	Config

F14 Profile

SF14 Recover

F15 Help

SF15 Extra

SF16 Exit

Note. When you press a function key to go to another screen, then return to the Network Status Summary screen, your current screen appears exactly as it did when last displayed: the same page is displayed, the same status items are selected, and the screen is either frozen or thawed. Context is preserved.

Function Key Notes

Table 3-7 describes the operation of some of the function keys and special requirements for using them.

Table 3-7. Function Key Notes (Network Status Summary Screen) *(page 1 of 2)*

Function Key	Special Requirements
F2	Causes an immediate update of the status items currently displayed on the screen; occurs even if the screen is in the frozen state.
F6/F7	Makes changes in the current status configuration file. To delete one or more status items from the configuration file, enter nonblank characters in column 1 for each item you want to delete, and press F6. Press F7 to add new status items into the file, either at the end or between two existing items. To add an item between two existing items, place the cursor on an item on the screen to indicate where you want the new item added. (The existing item on that line is pushed down to make room for your addition.) Then press F7, which displays the Status Item Configuration screen. Fill in the appropriate information, and press F12, which returns you to the Network Status Summary screen with the new item added in its desired position. To add an item at the end of the screen, position the cursor on the option line, and press F7. F6 and F7 also change the order of status items. First position the cursor on an item you want to move (or select it with a nonblank character), and press F6 to delete it. (A special buffer temporarily retains the most recently deleted item.) Then, move the cursor to the position where you want the item to appear and press F7. This displays the Status Item Configuration screen with the newest change. Press F12 to complete the operation.

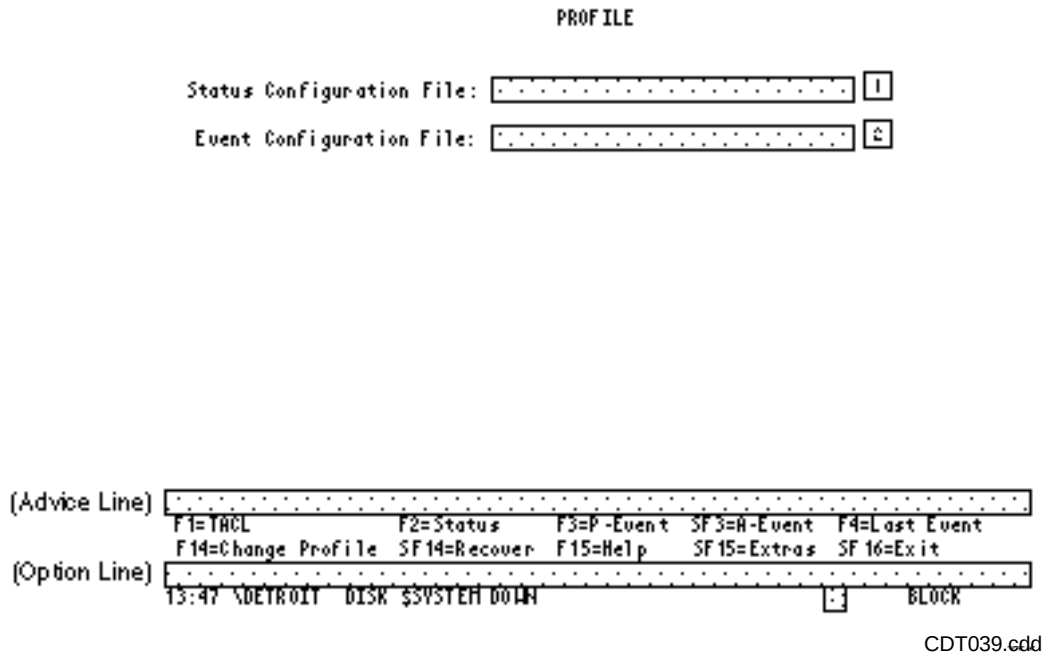
Table 3-7. Function Key Notes (Network Status Summary Screen) *(page 2 of 2)*

Function Key	Special Requirements
F9	Prints the entire screen—data and all headings and legends.
F10	Copies selected status items to the end of the clipboard file. Enter nonblank characters in column 1 preceding the items you want to copy, and press F10. You can print these selected status items by pressing F9.
F12	Has two different uses: • Displays the Status Item Configuration screen, if you select one or more status items for configuration (with a nonblank character in column 1). • Displays the Status Configuration screen, if you do not select any items. Use the Status Configuration screen when you want to change the screen update interval. If you select items, ViewPoint displays the Status Item Configuration screen for the first selected item. Then, make your changes and press F12 on that screen to advance from one item to the next. After you have configured the last selected item, press F12 to return to the Network Status Summary screen.

Profile Screen

Figure 3-9 shows the Profile screen.

Figure 3-9. Profile Screen Format



Field Descriptions

1. **Status Configuration File**. Name of the status configuration file that you are currently using. Select a different file by typing its name in this field, and press F14 (Change profile).
2. **Event Configuration File**. Name of the event configuration file you are currently using. Select a different file by typing its name in this field, and press F14.
3. **Function keys**

F1	TACL
F2	Status
F3	P-Event
SF3	A-Event
F4	Last Event
F14	Change Profile
SF14	Recover
F15	Help
SF15	Extras
SF16	Exit

Function Key Notes

Table 3-8 describes the operation of one of the function keys and special requirements for using it.

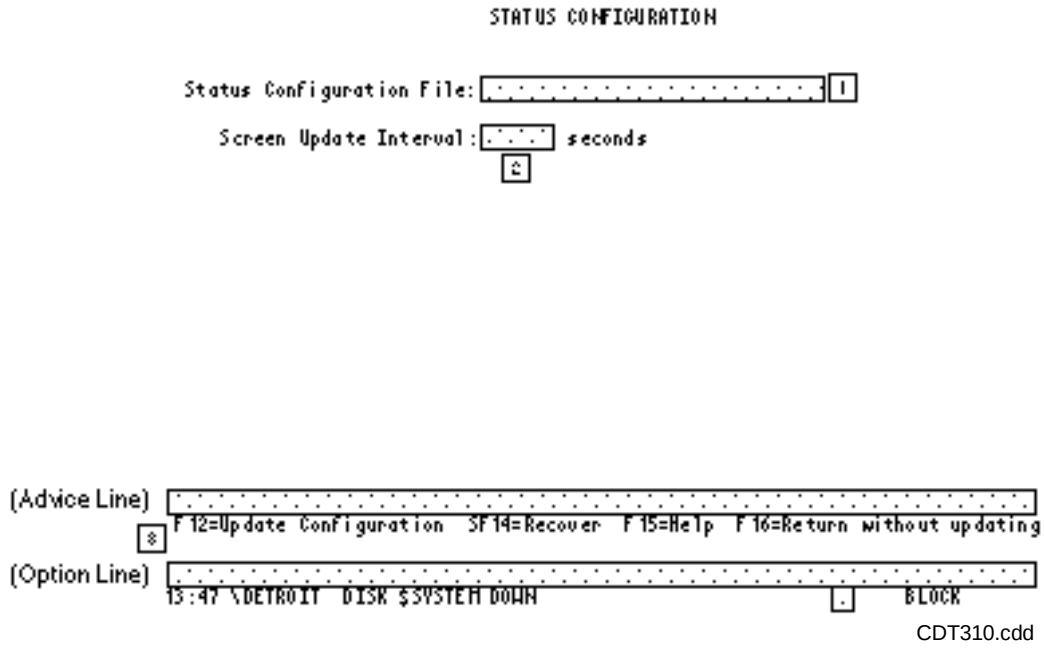
Table 3-8. Function Key Notes (Profile Screen)

Function Key	Special Requirements
F14	Reconfigures the logic for status or events (or both) according to parameters specified in the files named in the Status Configuration File or Event Configuration File fields (or both). The default files are typically your own ZZVPSTAT and ZZVPEVNT files; if these do not exist, ViewPoint uses the default files, STATDFLT and EVNTDFLT. If you make a change in these fields and press any function key, except F14, your change is ignored.

Status Configuration Screen

Figure 3-10 shows the format for the Status Configuration screen.

Figure 3-10. Status Configuration Screen Format



Field Descriptions

1. **Status Configuration File**. Name of the configuration file you are currently using. To change files, you need to use the Profile screen.
2. **Screen Update Interval**. Interval at which the status requests are automatically sent out to update the displayed status items. The default interval is 10 seconds.
3. **Function keys**

   F12 Update Configuration

   SF14 Recover

   F15 Help

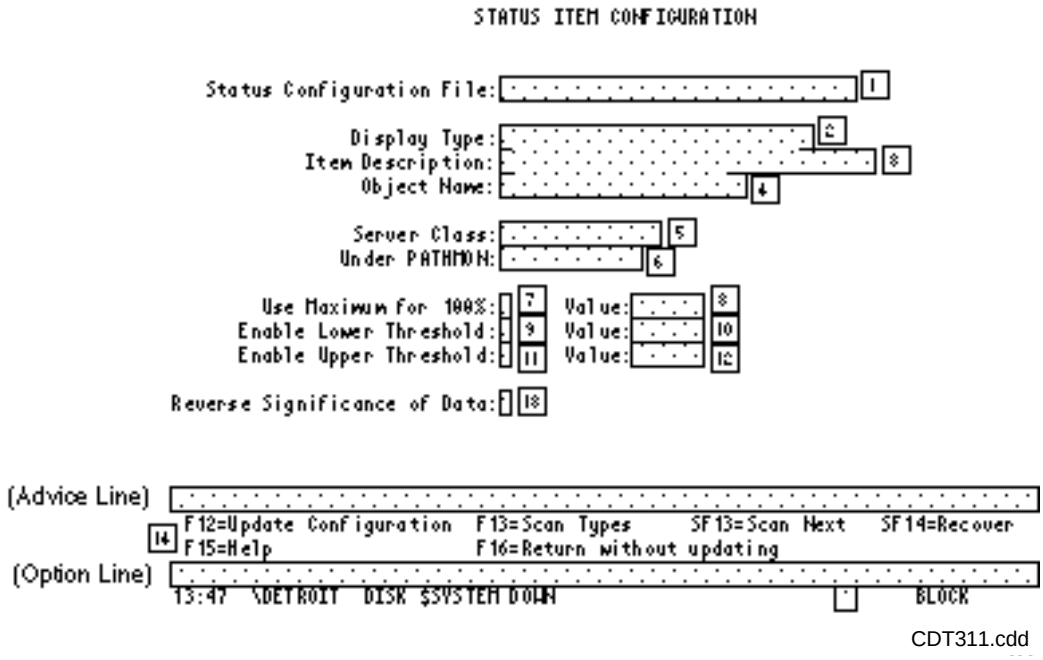
   F16 Return without updating

Note. Press F16 to return to the Network Status Summary screen without changing the update interval with any modifications you might have made. Press F12 to return to the Network Status Summary screen and cause a change to the update interval with your modifications.

Status Item Configuration Screen

Figure 3-11 shows the format for the Status Item Configuration screen.

Figure 3-11. Status Item Configuration Screen Format



Field Descriptions

1. **Status Configuration File**. Name of the status configuration file that you are modifying. Use the Profile screen if you want to modify a different Configuration file.

2. **Display Type**. Name by which the **status server** process identifies the status item being configured. In most cases, this field is filled in for you either by the status server itself (when you press F13 to scan the server types) or by the Network Status Summary screen when you requested configuration for a specific item. However, you can make an entry in this field to add or reconfigure a specific item if you know the valid display types for the given server (identified in field 5, Server Class).

3. **Item Description**. This is a 26-character field where you enter a description for the status item being configured. The description should be meaningful to you at your installation, using familiar node and object names. It need not be unique. It appears at the start of the line on the Network Status Summary screen.

4. **Object Name**. If the display type defined above consists of more than one object (such as the separate CPUs in a network node), you can make separate status items for each object by specifying an object name in this field. Use the name expected by the status server. The names expected by the standard status server are listed in

Table 3-9. Only certain display types can be broken down this way into separate objects.

Table 3-9. Object Names for Display Types

Display Type	Object Name
CPU-busy	0-15
Average-CPU-Busy	None
Disk-busy	Device name
Line-busy	Device name
PATHWAY-term	PATHMON process name
SNAX-lines-up	$SSCP
SNAX-PUs-up	$SSCP
SNAX-LUs-up	$SSCP
TMF-transaction-rate	None

5. **Server Class**. Name of the server class that is responsible for the display type defined in Table 3-3. Each status server class has its own collection of display types.
6. **Under PATHMON**. Normally this field is blank and the default name assumes the name of the server class that belongs to the Pathway system in which the current ViewPoint session is operating. However, you can obtain status from a server class that belongs to some other Pathway system; in that case, you must identify its PATHMON in this field, using the network form of the process name if it is a remote PATHMON.
7. **Use Maximum for 100%**. This is a one-character field where you enter a Y (yes) or an N (no), but only for status items that provide a count of objects. "Yes" means use the configured maximum count of objects as the basis for computing percentage on the Network Status Summary screen. "No" means use the value given in the next field as the basis. You might want to do this if a predictable number of objects are typically down or offline, or to improve resolution in cases where percentages are typically very low.
8. **Value**. If you entered N in the preceding field, specify your desired value in this field for the percentage computation basis.
9. **Enable Lower Threshold**. This is a one-character field where you enter a Y (yes) or an N (no). Entering a "Y" means highlight this line on the Network Status Summary screen when the computed percentage value falls below the percentage value stated in the next field (lower threshold value). "No" means ignore any lower threshold setting.
10. **Value**. If you entered Y in the preceding field, specify your desired value in this field for a lower threshold setting.

11. **Enable Upper Threshold**. This is a one-character field in which you can enter a Y (yes) or an N (no). Entering a “Y” means highlight this line on the Network Status Summary screen when the computed percentage value rises above the percentage value stated in the next field (upper threshold value). Entering an “N” means ignore any upper threshold setting.

12. **Value**. If you entered Y in the preceding field, specify your desired value in this field for an upper-threshold setting.

13. **Reverse Significance of Data**. This is a one-character field where you enter a Y (yes) or an N (no). Entering “Y” means consider the data as (total-n)/total. This makes it easy for you to reverse the meaning of a status item. For example, a “CPU Busy” item showing 40% could be a “CPU Idle” item showing 60%.

14. Function keys

F12	Update Configuration
F13	Scan Types
SF13	Scan Next
SF14	Recover
F15	Help
F16	Return without updating

Function Key Notes

Table 3-10 describes the operation of some of the function keys and special requirements for using them.

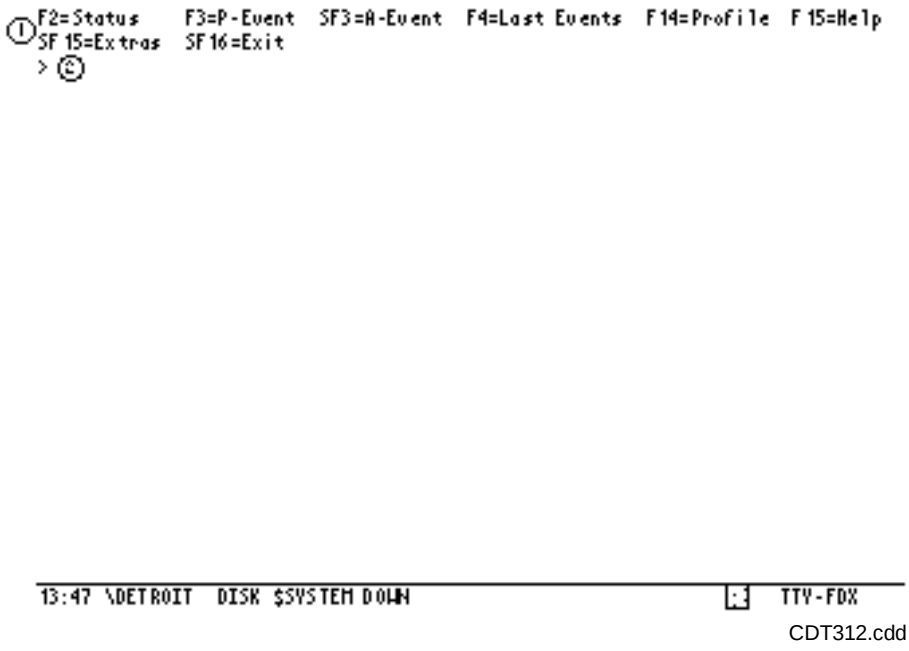
Table 3-10. Function Key Notes (Status Item Configuration Screen)

Function Key	Special Requirements
F12	Causes the current status-configuration file to accept the values on the screen. Depending on whether the item already exists, F12 either adds the described status item to the Network Status Summary screen or reconfigures the current status item with the new values. If you selected more than one item on the Network Status Summary screen, F12 advances you to the next item, displaying its current values. Pressing F12 multiple times advances you through all selected items, finally returning you to the Network Status Summary screen. To return immediately to the Network Status Summary screen, press F16.
F13	Retrieves the default values for display types associated with the status server class, usually in order to create a new status item. Has two ways to perform this retrieval: • Enter the name of a specific **display type** and server class in fields 2 and 5, then press F13. ViewPoint queries the specified server and, on locating the display type, fills in default values in other appropriate fields. You can then fill in other fields and modify default values. Press F12 to add this new status to the end of the status configuration file. • Leave the display type field (field 2) blank; enter the name of a specific server class in field 5 and press F13. ViewPoint retrieves the first display type for the specified server. You can then fill in other fields, or modify default values, and press F12. ViewPoint adds the configuration of the Status Screen to the default configuration file, Statdflt, for later use. Press SF13 to retrieve the next display type and modify it. After all the display types are retrieved, SF13 causes the first display type to be displayed again. You can use F16 to return to the Network Status Summary screen or use F13 to scan display types for some other status server class.
F16	Returns to the Network Status Summary screen without making any change to the configuration of the current status item.

TACL Screen

Figure 3-12 shows the TACL screen. This screen is not available in all ViewPoint manuals.

Figure 3-12. TACL Screen Format



Field Descriptions

1. Function keys

F2	Status
F3	P-Event
SF3	A-Event
F5*	DSM/PM
SF5*	DSM/PM
F4	Last Events
F14	Profile
F15	Help
SF15	Extras
SF16	Exit

∗This function key is active only if you have configured ViewPoint to run with DSM/PM. When you press F5 or SF5, the DSM/Problem Manager Menu displays on your screen.

Note. When the TACL interactive field becomes full, the function-key legend (lines 1 and 2) rolls off the top of the screen. To get the legend back, press F15 to display the Help screen, then press RETURN.

If you are using the RMI (Remote Maintenance Interface) terminal of a Compaq VLX system, you cannot use the SF16 function key to exit from the ViewPoint application. The RMI terminal uses SF16 to switch from a command interpreter (such as TACL) to the RMI main menu. To exit from the ViewPoint application, type :SF16 after the TACL prompt.

---

2. **TACL interactive field**. This is an unlimited number of lines for TACL commands and responses. You can access earlier pages with the PREV PAGE and NEXT PAGE keys (or PG UP and PG DN keys) and recall earlier commands from the TACL history buffer.

   All TACL capabilities are available (for more information, see the *TACL Programmer's Guide*). You can use TACL commands in this field to access your clipboard file and to use the Define Process functions. See Section 2, "Using ViewPoint," for examples of how to do these operations.

4 Process Definition Commands

Introduction

This section describes the syntax and semantics of the Define Process commands. The Define Process command library provides a set of commands to the TACL command interpreter. The library is available to you when you log on to your system in the same way as your other TACL library files.

Section 2, Using ViewPoint gives a brief introduction to using the Define Process capability. Section 5, Functional Description describes the terms and concepts that can help you understand the operation of the commands that are in this section. In particular, you might want to read the discussion under the heading "Network Command Functions."

Note. These commands can be used only if you are running ViewPoint with access to TACL.

Command Introduction

The commands in the Define Process library allow you to define, start, stop, and remove a process in any network node to which you have access. This means that you must have remote passwords established between your system and all other systems that you wish to access. Refer to the *Guardian User's Guide* for information on remote passwords.

You can supply input commands interactively, one line at a time, or from a file or variable. You can view or delete input and output, or save it in a file or in a variable.

Table 4-1 lists all the Define Process commands and gives a brief description of the function of each command. The remainder of the section provides complete descriptions and syntax for each command, and the syntax for invocation of a defined process. Table 4-1 and the complete command descriptions are listed alphabetically.

Table 4-1. Define Process Commands *(page 1 of 2)*

Command	Description
DP	Defines a background process, and unless otherwise indicated, starts the process.
_PCHECK	Returns information about defined processes that can be specified by a user-defined programmatic interface.
PHELP	Displays help text for the Define Process commands.
PINFO	Displays the status and characteristics of a background process.
POUT	Displays accumulated output from a defined process and removes the data from the queue.
PSHOW	Displays accumulated output from a defined process, or displays input that has not yet been sent to a defined process, and does not affect the status of the data in either queue.

Table 4-1. Define Process Commands *(page 2 of 2)*

Command	Description
PSTART	Starts a defined process.
PSTOP	Stops a defined process, leaving it in a stopped and defined state.
TOSS	Deletes accumulated output or unprocessed input from either an input or output queue.
UNDP	Removes a background process.
WAIT	Displays accumulated output from a background process and waits for the process to prompt for more input.
WAITREADY	Waits for a background process to prompt for more input.

△ **Caution.** You retain your local system and volume environment when you use Define Process, so make sure you use appropriate SYSTEM and VOLUME commands wherever necessary to avoid device location mistakes.

You should also be aware that Define Process does not do echo suppression to terminals. For example, when you log on and use PS MAIL, you normally do not see your password echoed back to the terminal; however, if you run PS MAIL as a defined process, your password is echoed on the terminal. This could be a security problem.

Define Process (DP) Command

The Define Process (DP) command defines a background process and enters the run parameters of the process in the Define Process directory. The DP command also starts the process. You have the option of defining the process, and not starting it, by using the NOSTART option (see NOSTART later in this section).

Note. TACL must be a named process in order for you to use the DP command.

```
DP program-file [ / option [ , option ]... / ]
                [ param-string ]
```

program-file

is the name of the file containing the object program to be run. Partial file names are expanded using the current TACL search list.

option

is any of the following options:

CPU *cpu-number*

assigns the CPU number of the processor where the defined process is to execute. Specify *cpu-number* as an integer in the range 0 through 15. If you

omit this option, the defined process executes in the same processor as your TACL, unless a $CMON dictates otherwise. (See the *Guardian Programmer's Guide* for information about $CMON processes.)

`DEBUG`

See the INSPECT option below.

`DEFMODE { OFF | ON }`

enables or disables the current TACL DEFINEs. It is the initial definition mode state for the new process. If omitted, TACL uses its current definition mode. Refer to the *TACL Reference Manual* for the current definition mode and for setting TACL DEFINEs.) See also Appendix E, "Supplemental Information for D-Series Operating Systems" in this manual.

`EXTSWAP`

specifies the volume and, optionally, the file name that is used for memory swaps of the process's extended data, if there is any. The EXTSWAP is the file used for swaps of the extended data segment during the execution of the process. If the file does not exist, it is created.

Refer to the *TACL Reference Manual* for more information about D-series systems. EXTSWAP is a RUN command option. See also Appendix E, "Supplemental Information for D-Series Operating Systems," in this manual.

`FKEY`

enables function key processing in **passthrough mode**. (Passthrough mode is described under the subsection, "Issuing Commands to a Defined Process," at the end of this section.) When a function key is pressed, the appropriate function key macro (such as F1 or F2) and the text of the command you typed just prior to pressing the function key passes to the function key macro. The function key macro receives the name of the defined process in the variable *`:dpprocess`*, and the text of the command in the variable *`:dpcommand`*.

`HIGHPIN { OFF | ON }`

enables a high-PIN process to run at low PIN. If HIGHPIN ON is specified, the process runs at high PIN, if it has been converted to run at high PIN. If the process has not been converted to run at high PIN, this option is not valid; it runs at low PIN. The default is ON.

The HIGHPIN option has several other uses. See the *TACL Reference Manual* section describing D-series systems. HIGHPIN is a RUN command option.

`INSPECT { OFF | ON | SAVEABEND }`

sets the debugging environment for the process being defined. Use ON and SAVEABEND to select INSPECT as the debugger; use OFF to select DEBUG. SAVEABEND is the same as ON except that it automatically creates a save file if the program ends abnormally (ABEND).

The INSPECT option also sets the debugging environment for any descendants of the defined process. See the *Inspect Manual* for more information.

`JOBID [ jobid ]`

sets the job ID for the new process.

`LIB [ file-name ]`

selects a user library file of object routines that is searched before the system library file to satisfy external references in the program being run. If you give the name of a library file, the program uses that library until you select another. The library file name is linked to the program file and remains in use for all runs of the program until you specify LIB without a file name. If you do not give a file name, LIB deletes the previous selection.

To run a program file with a user library, you must have write access to the program file; the library file name is written into the program's object-file header at run time.

To run the program again with the same library, you can omit the LIB parameter. To run the program again with no library (or with a different library), include LIB (or LIB *file-name*). See the RUN command definition in the *TACL Programmer's Guide* for more information about LIB.

`MACRO macro-name`

is the name of a macro to invoke each time this defined process is invoked (that is, by typing its name at a TACL prompt). DP passes the name of the defined process in the variable *:dpprocess,* and the text of the command in the variable *:dpcommand* to the macro. The macro is not invoked again if the same process is invoked directly or indirectly from within the macro.

`MEM num-pages`

is the maximum number of virtual data pages to be allocated for the defined process. Specify *num-pages* as an integer in the range 1 through 64. If you omit this option, or if *num-pages* is less than the compile-time value, the process uses the compilation value.

`NAME [ $process-name ]`

part of the process-pair directory (PPD) is an operating system name that you can assign to the defined process. Specify *process-name* as an alphanumeric string of one to five characters, the first of which must be alphabetic. (For network access, the name must be no more than four characters.) If you omit this option, the defined process is not named (to the operating system) and has only a process ID (a CPU number and process number). If you include NAME without $*process-name*, the operating system generates a name for the defined process. The name of the process appears in the destination control table (DCT).

NOHISTORY

disables history processing for commands that are interactive with the process. If history processing is enabled, each command passed to the process in interactive mode is saved in the TACL history buffer. In addition, previous commands can be fixed with FC, invoked again with !, or viewed again using ? at the interactive prompt. Note that FC processing is trapped by TACL, not by the process. The current history number (surrounded by hyphens) appears to the left of the process's prompt.

NOSTART

disallows starting the process after the process is defined. The process is started automatically when the first command is sent to the process or when you issue a PSTART command to it.

NOWAIT

sets that the default mode for commands to the process is NOWAIT. If you omit this option, the default mode is WAIT.

PNAME *process*

assigns a unique symbolic name to the defined process. This is different from the Compaq *NonStop*™ Kernel $*process-name*. Use this name to access the process once it is defined. The name you choose should not coincide with the name of a variable you have pushed with #PUSH. If this option is omitted, DP assigns the name of the program file name to the defined process. The process name can be 1 to 32 characters long.

PREPARSE *macro-name*

defines the name of a macro that parses each command line before it is sent to the process. The macro is passed the name of the defined process in the variable :*dpprocess*, and the text of the command in the variable :*dpcommand*. The macro that was parsed can scan the typed command, alter it, mask it, or provide other services.

PRI *priority*

sets the execution priority of the defined process. If you omit this option, the defined process is given a priority of your current TACL, less one (unless a $CMON process dictates otherwise). Refer to the *Guardian Programmer's Guide* for information about $CMON processes. Specify *priority* as an integer in the range 1 through 199.

QUICK

forces future PSTART commands to not wait while the process starts up, and returns a command input prompt after sending the startup message to the defined process. Normally, PSTART waits for the process to prompt before completing.

This option also causes the process's banner, if present, to show up in the output variable of the process after startup. If this is a problem, use the TOSS command to clear the output queue before running the process.

`QUIET`

forces future PSTART commands not to print a message when starting the defined process. If this option is omitted, PSTART prints a starting verification message when it is starting the process. (The message can be useful to see when a process is being recreated after being stopped for any reason—such as an unintentional command error.)

`SHOWPROMPT`

echoes that the prompt of the defined process to the terminal when a command is issued to the process out of passthrough mode. Normally, the prompt is not seen in this case, but there are certain programs (such as continuation prompts in PS MAIL-TTY) in which it might be important to see the text of the prompt.

`SWAP [ `*`file-name`*` ]`

specifies the name of the file that holds the virtual data of the process. When a process is executing, the system allocates a temporary file on the same volume as the program file for swapping the data stack. When the process terminates, the temporary swap file is automatically purged. However, if the swap file has a permanent name, the swap file is not purged.

With the SWAP option, you can:

- Specify a permanent file name—the file contains an image of the data stack when the process terminates.
- Specify a different volume for the swap file (by specifying only a volume name)—useful when the program file's volume is full or busy.

The SWAP option also specifies the default volume for extended data segments. See the *Guardian Programmer's Guide* for more details.

`TERM [ \`*`system`*`.]$`*`terminal-name`*

specifies the home terminal for the defined process. If you omit this option, the defined process uses your TACL home terminal. For $*terminal-name*, specify a valid name for a terminal or process. That is, follow the dollar sign with an alphanumeric string of one to six characters, the first of which must be alphabetic. For remote access, you must use five or fewer characters after the dollar sign.

`param-string`

is a program parameter or a string of parameters sent to the defined process in the startup message. Leading blanks and trailing spaces are deleted. Square brackets and vertical bars cannot be included.

Considerations

- To resolve the program name for the file you specify, DP uses the current setting of your TACL #PMSEARCHLIST. This is unlike the TACL command, RUN, which uses your current #DEFAULTS to resolve the program name for your specified file.
- The default symbolic name of a defined process is the file-name portion of the process program file name. For example, this command defines a PERUSE process with the symbolic name of PERUSE:

  ```
  > DP PERUSE
  ```

  It might not be desirable to use a symbolic name for a defined process that is the same as the name of a utility, such as PERUSE, since it could interfere with your ability to run the program normally. In the above example, the program PERUSE would have to be run with the following command:

  ```
  > $SYSTEM.SYSTEM.PERUSE
  ```

  The better solution would be to use the PNAME option to assign some different symbolic name to the defined process.

△ **Caution.** Be aware that a collector process for the Spooler can have only 32 opens at a time. If you define PERUSE processes, as in the above examples, and leave them running, you are diminishing the effectiveness of the Spooler to process other jobs.

- To run a process on a remote system, specify *system-name* before the name of a program file. For example, this command defines an EDIT process on the \CHICAGO system:

  ```
  > DP \CHICAGO.EDIT
  ```

  The DP command assumes that the process is on a remote system and that the volume and subvolume that contain the program file are $SYSTEM.SYSTEM.
- The MACRO option allows you to keep the background process coordinated with your environment; for instance, you can send a VOLUME command to FUP before issuing a command. The process name is passed to the macro, so you can have an expert macro coordinate all your defined processes that need this service.

Examples

The sample sequence of commands in Figure 4-1 defines and starts two processes: one in the local system (command line 1) and one in a remote system (line 6). Note that these defined processes are invoked simply by typing their symbolic names in response to a TACL prompt. (The complete syntax for invoking a defined process is given at the end of this section.)

Figure 4-1. Examples of Using the DP Command

```
1> DP FUP /PNAME FU,MACRO VOLUMER/ ==define the first process
pstart: starting \HOME.$SYSTEM.SYS01.FUP process
2> FU                              ==invoke the process
VOLUME $DATA.MYSUBV
-3-- INFO *                        ==issue a local FUP command
                 CODE    EOF     LAST MODIF    OWNER   RWEP
$DATA.MYSUBV
 MYKEYS           101   2536   16JUL86 13:01    8,16   CUCU
 MYMACS           101   2194   17JUN86 09:35    8,16   CUCU
 TACLCSTM         101   3729   05AUG86 10:11    8,16   CUCU
 TEDPROFL         115  20480   24SEP86 15:20    8,16   CUCU
-4-- [CTRL-Y]                          ==return to TACL
4> SYSTEM \EASTUS                      ==set remote system default
5> VOLUME $KEEP.TODAYS                 ==set remote volume default
6> DP EDIT /PNAME EDEDIT/              ==define the remote process
pstart: starting \EASTUS.$SYSTEM.SYSTEM.EDIT process
7> EDEDIT SCHED R                      ==invoke the remote process
 TEXT EDITOR - T9601B30 - (01MAR86)
 CURRENT FILE IS \EASTUS.$KEEP.TODAYS.SCHED
* LIST 1/5                            ==issue a remote EDIT command
        1
        2
        3  Today's Schedule – East USA
        4
        5
* [CTRL-Y]                                    ==return to TACL
8>
```

The volumer macro used in the above example is as follows:

```
?section volumer text
  ==This macro provides the volume command for any background
  ==process that declares it as its "macro."

#append [_pcheck/inputv/ [:dpprocess]] volume [#defaults
/current/]
```

When you press the CTRL-Y key, you return to your local TACL environment (lines 4 and 8). Both defined processes in this example keep running, and they are expecting a response to the latest prompt. You can return to the environment of either one of these defined processes by typing the symbolic name of the process—as at command lines 2 and 7.

_PCHECK Command

This command is provided for programmatic use, not interactive use; its definition is included here for your information. _PCHECK collects information about defined processes and expands this information to a space-separated list of the information requested. The PINFO command, defined later, provides a subset of the information

that _PCHECK provides. Refer to Section 6, Customizing ViewPoint for further details and examples of the use of this command.

Note. If your TACL use list is not set up to include :utils:DP, you must precede the _PCHECK command with :utils:dp.

```
_PCHECK [ / option [ , option ]... / ] [ process-list ]
```

option

is any of the following options:

EXISTENCE

returns -1 if the process is defined; otherwise, it returns 0.

FILE

returns the IN and OUT file names of the process, if the process is running.

FKEY

returns -1 if function-key processing is enabled; otherwise, it returns 0.

HISTORY

returns -1 if command history is enabled; otherwise, it returns 0.

INPUTLINES

returns the number of lines in the process's input queue.

INPUTV

returns the name of the process's input variable.

MACRO

returns the name of the process's macro, if defined.

OUTPUTLINES

returns the number of lines in the process's output queue.

OUTPUTV

returns the name of the process's output variable.

PNAME

returns the symbolic name assigned to the process.

PREPARSE

returns the name of the process's preparser, if defined.

PROCESSID

returns the operating system name of the process, if the process is running.

PROGRAMFILE

returns the file name of the program being executed.

QUICK

returns -1 if the process is restarted without waiting for the prompt; otherwise, it returns 0.

QUIET

returns -1 if the confirmation message is printed when starting the process; otherwise, it returns 0.

READY

returns -1 if the process is waiting for more input; otherwise, it returns 0.

RUNNING

returns -1 if the process is running; otherwise, it returns 0.

RUNOPTS

returns the RUN options used to start the process.

SHOWPROMPT

returns -1 if the prompt is shown after each one-line command; otherwise, it returns 0.

STARTUP

returns the contents of the startup message.

STATUSV

returns the name of the process's status variable.

WAIT

returns -1 if the default wait mode is wait; 0 if the mode is nowait.

process-list

is a list of legal, unique symbolic names you have assigned to defined processes. If you specify an asterisk (*), information for all your defined processes is displayed.

Considerations

If you specify more than one request option, the pieces of information are separated by spaces and are listed in the same order as the requests. If you specify more than one process name, the groups of information for each process are separated by spaces and are listed in the same order as the requests.

Information is returned for previously-defined processes. If the process is not defined, 0 is returned for the EXISTENCE option, and nothing is returned for the other options.

PHELP Command

The PHELP command provides online help for the Define Process commands.

Note. If your TACL use list is not set up to include :utils:DP, you must precede the PHELP command with :utils:dp.

```
PHELP [ command... ]
```

command

is any of the following commands:

```
ALL
COMMANDS
DP
_PCHECK
PHELP
PINFO
POUT
PSHOW
PSTART
PSTOP
TOSS
UNDP
WAIT
WAITREADY
```

ALL

provides help for all commands.

COMMANDS

provides help for issuing commands to a process.

All other *commands* are names of Define Process commands. The appropriate help message is provided. If no *command* is supplied, a command summary is provided.

PINFO Command

Use the PINFO command to display the status and characteristics of one or more defined processes.

Note. If your TACL use list is not set up to include :utils:DP, you must precede the PINFO command with :utils:dp.

```
PINFO [ / DETAIL / ] [ process-list ]
```

DETAIL

gives detailed information on status and characteristics of the specified defined processes. See Figure 4-3 for the display format provided by the DETAIL option.

process-list

is a list of unique, symbolic names that you have assigned to defined processes. If you do not give a process name (or if you specify an asterisk (*)), information for all your defined processes is displayed.

PINFO Listing Formats

Figures 4-2 and 4-3, below, are examples of the listing formats for the PINFO command. Note that both formats provide the symbolic process name, the program file name, line counts for both input and output queues, and the readiness state of the process. The detail format additionally provides the operating system process name (or cpu and pin numbers), the in and out file name for the process, the current default wait mode, the setting of the HISTORY option, and the setting of the SHOWPROMPT option.

In these two figures, NAME is the symbolic name you have assigned to the process, and PROGRAM is the file portion of the program file name. If the program is being run remotely, the file name is prefixed with the system name.

The INPUT and OUTPUT columns list the number of lines of text in the input and output queues, respectively. If either of these queues is empty for a given process, the word EMPTY is displayed.

Figure 4-2. PINFO Listing Without Detail

```
NAME        PROGRAM      INPUT      OUTPUT      READY

FLUP        FUP          EMPTY      EMPTY       YES
P           PERUSE       EMPTY      EMPTY       YES
PS          PSMAIL       EMPTY      EMPTY       YES
EDEDIT      EDIT         EMPTY      EMPTY       YES
```

Figure 4-3. PINFO Listing With Detail

```
NAME      PROGRAM   PROCESS          FILE           INPUT    OUTPUT   READY

FLUP      FUP       2,82             $F108.#S13     EMPTY    EMPTY    YES
P         PERUSE    6,24             $F108.#S12     EMPTY    EMPTY    YES
PS        PSMAIL    7,104            $F108.#S15     EMPTY    EMPTY    YES
EDEDIT    EDIT      \EASTUS.4,64     $F108.#S11     EMPTY    EMPTY    YES

WAIT      HISTORY     PROMPT

YES       YES         NO
YES       YES         NO
YES       YES         NO
YES       YES         NO
```

In the READY column, YES is displayed if the process is running and ready to accept commands; if not, NO is displayed. In the WAIT column, YES is displayed if the default command mode is WAIT; if the mode is NOWAIT, NO is displayed.

The HISTORY and PROMPT columns display YES if these options are enabled and NO, if they are not enabled.

POUT Command

The POUT command displays accumulated output from a defined process. Output is removed from the output queue as it is being displayed. You can use POUT to view output text after issuing a NOWAIT command, or after pressing the BREAK key during output.

Note. If your TACL use list is not set up to include :utils:DP, you must precede the POUT command with :utils:dp.

```
POUT [ / out-option / ] process-list
```

out-option

is any of the following options:

OUT *file-name*

names the file to which the output should be directed. If the file does not exist, it is created.

OUTV *variable*

names an existing variable to which the output should be directed.

process-list

is a list of unique, symbolic names you have assigned to defined processes. If you specify an asterisk (*), information for all your defined processes is displayed.

Considerations

If you press the BREAK key during output and the process is prompting, the portion of the output viewed since the process prompted is not removed from the output queue.

PSHOW Command

The PSHOW command displays accumulated output from a defined process or input that is yet to be sent to the process. The display can be sent to a specified file or variable instead of to your terminal. Neither the output queue nor the input queue is affected by this command. You can use PSHOW to view input or output text after issuing a NOWAIT command or after pressing the BREAK key during output.

Note. If your TACL use list is not set up to include :utils:DP, you must precede the PSHOW command with :utils:dp.

```
PSHOW [ / out-option [, queue-option ] / ] process-list
```

out-option

is any of the following options:

OUT *file-name*

names the file to which the display output should be redirected. If the file does not exist, it is created.

OUTV *variable*

names an existing variable to which the display output should be redirected.

queue-option

is any of the following options:

INPUT

shows the contents of the input queue.

OUTPUT

shows the contents of the output queue. This is the default.

process-list

is a list of unique, symbolic names you have assigned to defined processes. If you specify an asterisk (*), information for all your defined processes is displayed.

Considerations

PSHOW does not remove text from the input or output queue as POUT does. You would typically use PSHOW Command to monitor a process's progress by looking at

the input to see how many commands are left to be executed by the process or by looking at the output to see how much output has been produced.

PSTART Command

The PSTART command starts a process if the process is not already running. If the process cannot be started, a message is displayed.

Note. If your TACL use list is not set up to include :utils:DP, you must precede the PSTART command with :utils:dp.

```
PSTART process-list
```

process-list

is a list of unique, symbolic names you have assigned to defined processes. If you specify an asterisk (*), information for all your defined processes is displayed.

Considerations

- PSTART is automatically invoked whenever a command is issued to a defined process that is in the stopped state. You normally do not have to use PSTART to start a process unless you want to make sure the process is ready for commands before the process is needed.
- The process is started with the current user ID. If you change user IDs after the process is started, the process continues to run under the old user ID. If this is a security problem, the LOGON routine can be changed to stop all defined processes before changing user IDs.
- Input and output queues are preserved when a process is being restarted.

PSTOP Command

The PSTOP command stops a process that is currently running. If the process cannot be stopped, a message is displayed.

Note. If your TACL use list is not set up to include :utils:DP, you must precede the PSTOP command with :utils:dp.

```
PSTOP process-list
```

process-list

is a list of unique, symbolic names you have assigned to defined processes. If you specify an asterisk (*), information for all your defined processes is displayed.

Considerations

- You can use the PSTOP command if your current user ID is included for the permissions of this process.
- Removing a defined process automatically stops the process.

TOSS Command

The TOSS command deletes accumulated input or output from an input or output queue of the process. You can use TOSS to discard unwanted input or output after issuing a NOWAIT command or after pressing the BREAK key during output.

Note. If your TACL use list is not set up to include :utils:DP, you must precede the TOSS command with :utils:dp.

```
TOSS [ / queue-option / ] process-list
```

queue-option

is any of the following options:

INPUT

deletes contents of the input queue.

OUTPUT

deletes the output queue contents. This is the default.

process-list

is a list of unique, symbolic names you have assigned to defined processes. If you specify an asterisk (*), information for all your defined processes is displayed.

Considerations

Using the TOSS Command to delete the input queue contents does not delete an EOF pending for the process.

UNDP Command

The UNDP command stops and undefines one or more defined processes. A process is undefined by popping (#POP) all of the variables that were pushed or defined to manage it and removing the process entry from the Define Process directory.

Note. If your TACL use list is not set up to include :utils:DP, you must precede the UNDP command with :utils:dp.

```
UNDP process-list
```

process-list

is a list of unique, symbolic names you have assigned to defined processes. If you specify an asterisk (*), information for all your defined processes is displayed.

Considerations

- Once a process is undefined, all attributes of the process are lost. To restart the process, you must define it again using DP.
- UNDP issues the PSTOP command to stop a process. If the process is not stopped, then the process is already undefined or you do not have permission to remove the define process.

WAIT Command

The WAIT command displays accumulated output from a defined process and waits for the process to prompt. Output is removed from the output queue as it is being displayed. Instead of sending to your terminal, you can send the display output to a file or a variable. WAIT also allows you to view process output after issuing a NOWAIT command or after pressing the BREAK key during output. If you want to return to the TACL prompt before output is complete, press the BREAK key.

Note. If your TACL use list is not set up to include :utils:DP, you must precede the WAIT command with :utils:dp.

```
WAIT [ / out-option / ] process-list
```

out-option

is any of the following options:

OUT *file-name*

names the file to which the output is redirected. If the file does not exist, OUT creates it.

OUTV *variable*

names an existing TACL variable to which the output is redirected.

process-list

is a list of unique, symbolic names you have assigned to defined processes. If you specify an asterisk (*), information for all your defined processes is displayed.

Consideration

WAIT is invoked automatically during commands using the WAIT option.

WAITREADY Command

The WAITREADY command simply waits for the specified process to prompt. Output from the process is preserved and not displayed. You can use WAITREADY after issuing a NOWAIT command to synchronize with the process. If you change your mind and want to return to the TACL prompt before the process prompts, press the BREAK key.

Note. If your TACL use list is not set up to include :utils:DP, you must precede the WAITREADY command with :utils:dp.

```
WAITREADY process-list
```

process-list

is a list of unique, symbolic names you have assigned to defined processes. If you specify an asterisk (*), information for all your defined processes is displayed.

Considerations

- WAITREADY is most useful within macros for synchronizing with the defined process after issuing a series of NOWAIT commands. The output is not displayed during a wait, so it can be manipulated after the process completes and you see the prompt.
- WAITREADY returns control if the process stops. This is different from PINFO ready state, where the process is running and prompting.

Issuing Commands to a Defined Process

You can send commands to a defined process by invoking the symbolic name of the process.

```
process [ / in-option [ , out-option ] / ] [ command-text ]
```

process

is the unique symbolic name you have assigned to the defined process.

in-option

is any of the following options:

BLANK

sends the process a blank line.

EOF

sends the process an EOF.

IN *file-name*

takes input from the specified existing file.

INV *variable*

takes input from the specified existing variable.

out-option

is any of the following options:

OUT *file-name*

waits for the process to finish and then directs its output to the named file. If the file does not exist, it is created.

OUTV *variable*

waits for the process to finish and then directs its output to the named existing variable.

NOWAIT

waits for the process to finish or displays output. This option overrides the default wait mode of the process.

WAIT

waits for the output from the command displayed on the terminal. This option overrides the default wait mode of the process.

command-text

is a one-line command to pass to the process. The syntax is analogous to the text you enter following a TACL RUN command.

Passthrough Mode

If *command-text* is omitted and input has not been redirected to a file or variable, you enter passthrough mode with the process. In this case, the process prompts you; the prompt is prefixed with the current history number (unless history is disabled). Each command you type at the prompt is passed literally to the process (if not modified by a preparse routine), and the terminal waits or does not wait for output from the command, depending on the current wait mode.

If history is enabled, the FC, !, and ? commands are trapped and processed by TACL. History text appears in the TACL history buffer. You can view this buffer with the HISTORY command at a TACL prompt.

If function-key processing is enabled, function keys are trapped. The function-key macro is invoked (such as F1 or F2), passing the commands typed before the function key was pressed.

To exit from passthrough mode and return to the TACL prompt, type CTRL-Y or BREAK.

Considerations

- The BREAK key is processed by TACL and cannot be passed to a defined process. If a process depends on the BREAK key to cancel processing, you can stop processing only by stopping the process. If a process produces output in which you are not interested, use the TOSS command to delete the output.
- If a command is issued a NOWAIT or the BREAK key is pressed during output, use the WAIT, POUT, or PSHOW commands to see the output.
- If history processing is enabled for a process, any FIX command text (such as for the EDIT command, FIX, and not the FC command) that is processed by the process and not by TACL, is offset to the left by the number of characters in the history number plus two for the surrounding dashes. This is a side effect of the fact that TACL is prompting your terminal on behalf of the process and the process does not know that TACL has inserted the history number into its prompt. To compensate for this offset, align your cursor with the portion to be fixed, press the space bar once for every digit in the current history number, then press it twice. You might wish to disable history (NOHISTORY) with such a process.
- Some command interpreters, such as FUP, use the record length returned from DEVICEINFO for the output record length (132 characters for a defined process) since the OUT file is the TACL process. This causes the list of files returned from a FUP FILES command to be folded at 80 characters and not formatted for the terminal. To correct this problem, use a REPORTWIDTH 80 command in FUP. For TACL and PERUSE, use #WIDTH and NUMCOL, respectively, to set the actual record length to 80.
- One EOF can be pending for a process. All other forms of process input can be stacked to any level.
- Since you can send an EOF to a process only with the command *process*/EOF/, where *process* is the current process, you should define a function key to issue this command to the current process. This allows you to send an EOF to the process while in passthrough mode.
- Before a command is issued to a process, the process is restarted, if necessary. If the process stops while issuing NOWAIT commands, the fact that it stopped is not detected, and the process is not restarted. The commands are put on the process's input queue to be read when the process is restarted.

5 Functional Description

The first three figures in this section provide an overview of the ViewPoint architecture. The remainder of the section illustrates and discusses three major aspects of the application—TACL (command and control), network status, and event presentation. The section also introduces all significant terms relevant to the ViewPoint application and shows their functional context.

This section describes how ViewPoint works. For a step-by-step description of how to use ViewPoint to perform particular tasks, read Section 2, Using ViewPoint.

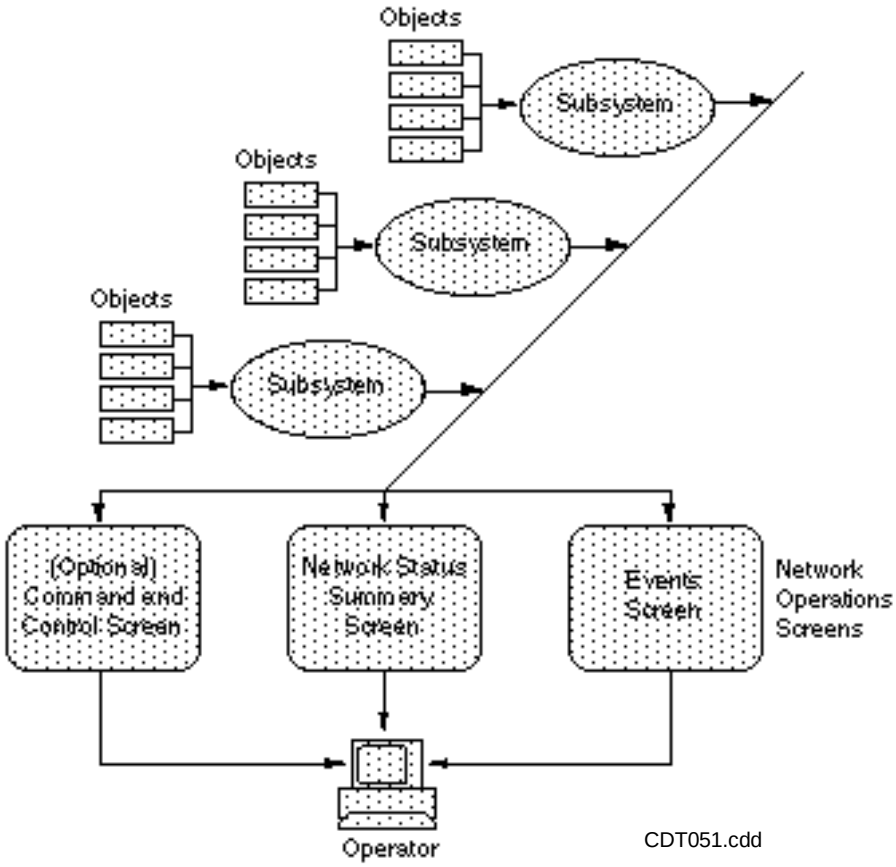
Overview of ViewPoint Architecture

The ViewPoint application provides three basic operating screens. These permit three kinds of activities with respect to objects controlled by the subsystems in a system or a network of systems. Figure 5-1 illustrates this concept, showing the three screens: command and control (TACL), network status, and events. You can select any one of these screens with function keys on the terminal keyboard.

Note. Some ViewPoint applications run on dedicated terminals without direct access to the TACL command and control screens.

The subsystem processes shown in Figure 5-1 can be Compaq-provided processes, such as PUP, FUP, TMF, or Pathway, or they can be processes provided and integrated into the application by your own network application programmers. Some examples of the objects referred to in Figure 5-1 are CPUs, disks, terminals, and communications lines.

Figure 5-1. Operator View of ViewPoint



The TACL screen (if available) provides all the standard features of the TACL product, plus the set of commands constituting the Define Process feature. This feature allows you to define (and instantly switch among) various processes anywhere in the network. In addition, you can switch to any of the other ViewPoint screens without exiting from any of the defined processes.

The Network Status Summary screen provides various status reports about selectable objects in the network. For example, a report might be about units started or units busy.

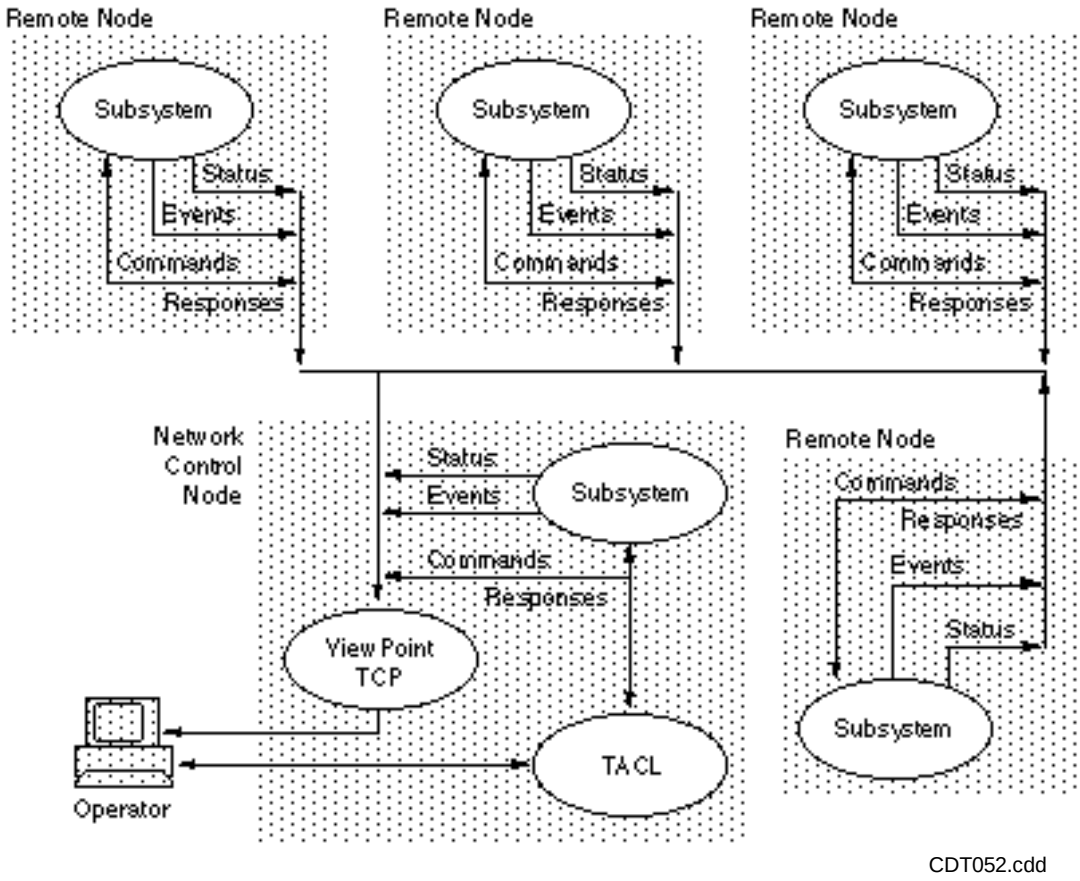
The Events screen provides a means of viewing current or past event messages that have been generated by the various subsystems. The messages can be for such functions as errors or failures, warnings, and requests for operator actions. Certain kinds of messages can be designated as critical and are highlighted on the screen. Action event messages also are highlighted.

Figure 5-2 shows a simplified conceptual arrangement for configuring a network to be controlled by ViewPoint.

The central controlling process of the application is the ViewPoint TCP (terminal control process). The network node on which TCP is running is called the **network**

control node (NCN). Commonly, network operators log on at this node, although it is possible to log on remotely.

Figure 5-2. Network Presentation for Multiple Nodes



There can be more than one NCN in a network, allowing various operators to share responsibilities or to have different levels of responsibility in network control. Figure 5-2 assumes one NCN and four remote nodes. One operator is logged on at this NCN.

In Figure 5-2, a representative subsystem is shown in all five nodes (including the NCN). These processes all interact with the ViewPoint TCP to allow reporting of events and status to the operator. Your TACL process is responsible for command and response interaction with these five subsystems.

Figure 5-3 shows the basic architecture for all three logical portions of the ViewPoint application—TACL, status logic, and events logic. The figure shows two separate network nodes to illustrate the typical distribution of processes in the network environment.

ViewPoint TCP

The ViewPoint TCP is the terminal control process portion of a Pathway application (see *Introduction to Pathway*). The TCP has three major functions: terminal management, code management, and message management.

The terminal management function routes information to and from each logged-on operator. The code management function manages the multithreaded execution of the network application programs (SCREEN COBOL programs) for each operator. The message management function communicates with the remainder of the system.

The operator default subvolume contains your TACLCSTM file, your clipboard (ZZVPCLIP) file, and various other files that your user ID owns.

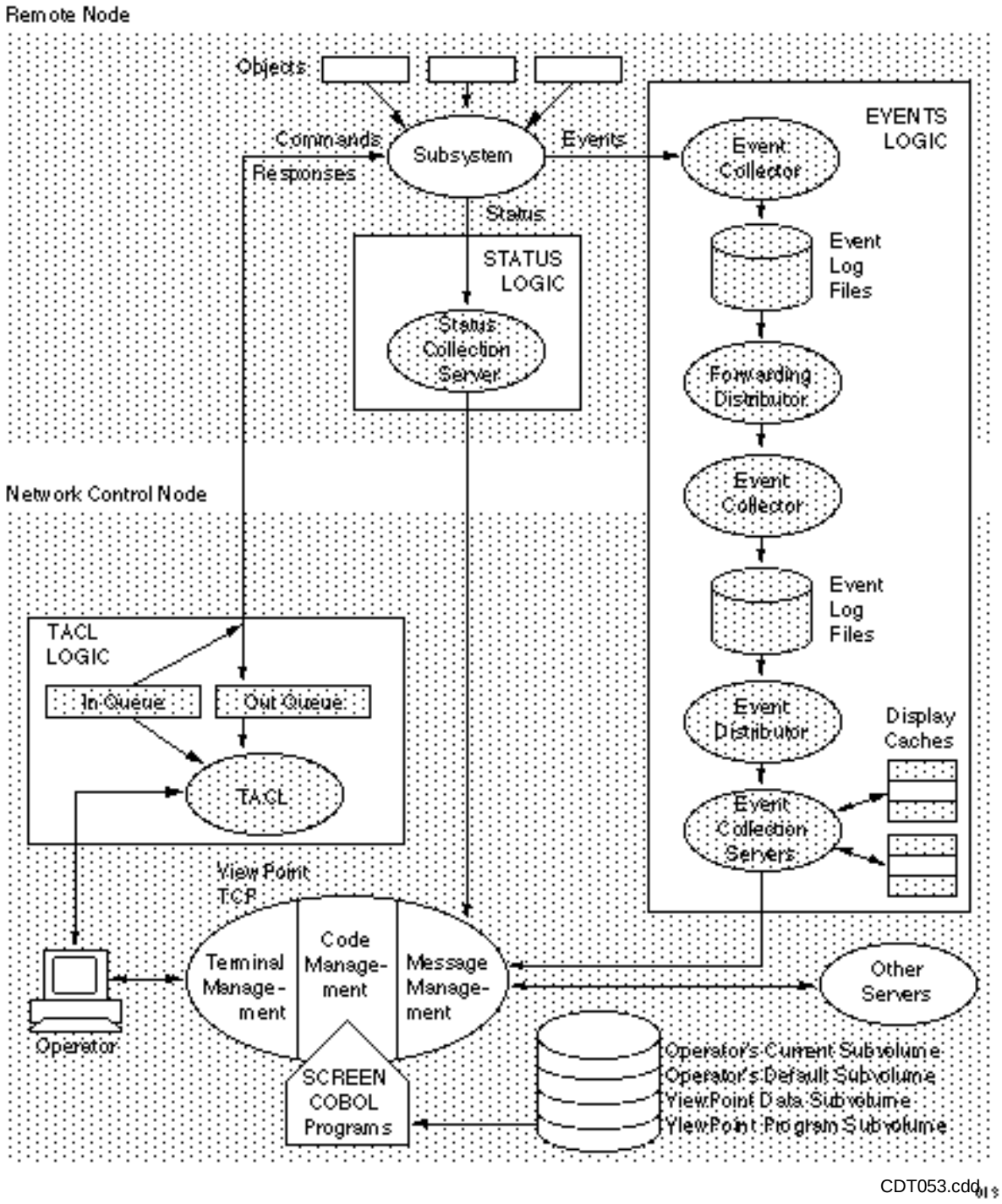
The ViewPoint program subvolume contains SCREEN COBOL programs, various components of Pathway and TACL, and other files that all operators must be able to access. The ViewPoint data subvolume holds the display caches for events and status.

The remaining paragraphs under this subheading separately discuss each of the three logical parts of the application—TACL, status logic, and events logic.

TACL

When your operator terminal is in TACL mode, the terminal communicates directly with your TACL command interpreter process. The ViewPoint TCP is effectively disconnected from your terminal, even though servers can be actively performing event functions for you, and action and critical events are reported on the twenty-fifth line of the screen.

Figure 5-3. Basic Network Presentation Architecture



TACL commands are transmitted to a particular subsystem by way of the TACL in queue, and responses are received in the TACL out queue. You can manipulate both queues with Define Process commands.

Status

Status reports on a selected set of objects are obtained by switching to the Network Status Summary screen. After you switch to this screen, the status display is updated automatically at intervals that you can specify, or you can update the display manually by pressing a function key.

The status request is issued by one of the SCREEN COBOL programs running under control of the ViewPoint TCP. This program forms a number of operating system messages that are sent to status collection servers in one or more network nodes. Each message is a request for one item (of up to 16 on a page). The status collection servers forward these requests to the subsystems controlling the objects to be reported on that page.

As status items are returned, they are received by the SCREEN COBOL program in the ViewPoint TCP. When the update interval is complete for updating the Network Status Summary screen, all items that are ready are displayed on your terminal.

The ViewPoint TCP runs a separate SCREEN COBOL program thread for each logged-on operator, so this process can simultaneously manage status reporting for any number of network operators.

Events

Notifications of events from subsystems are received by an event collector for a given network node. The collector records notifications in an event log file as event messages. A forwarding **distributor** in that node forwards the messages to the event collector in the NCN for recording in the NCN event log file—along with events from various other nodes. A ViewPoint **event collection server** puts the messages into a display cache, such as the primary-events cache (for all operators) or into an alternate-events cache (for a particular operator).

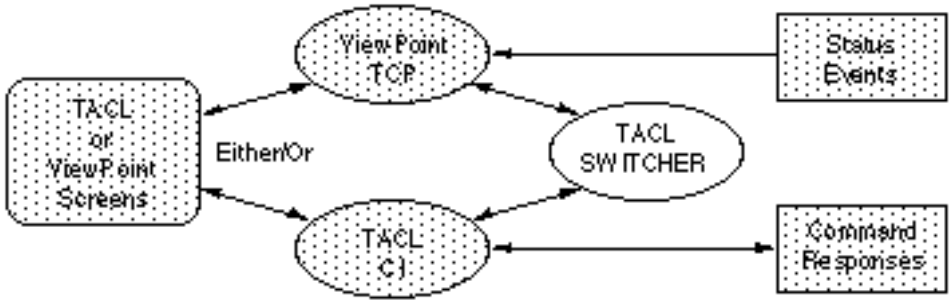
When you give a request from the Primary or Alternate Events screen, you can display on your terminal a page of 16 events. These events are read from a cache.

Network Command Functions

The next four figures show the network command functions available when you use a TACL command interpreter from ViewPoint. Figure 5-4 illustrates switching between the ViewPoint TCP and the TACL command interpreter. Either of these two processes controls your terminal and displays either a TACL or ViewPoint screen. A process called the TACL switcher manages the switchovers between the TACL command interpreter and the ViewPoint process. The operation is described in the next paragraph.

Assume you have been using the ViewPoint screens (for status or event functions), and you now wish to switch to the TACL screen. You press the TACL function key (F1). This action causes the ViewPoint process to send a message to the TACL switcher. The message says your task in the ViewPoint process is done and that the TACL switcher should activate the TACL process. The TACL switcher activates the TACL command interpreter, which assumes control of your terminal and suspends your task in the ViewPoint process.

Figure 5-4. Role of the TACL Switcher

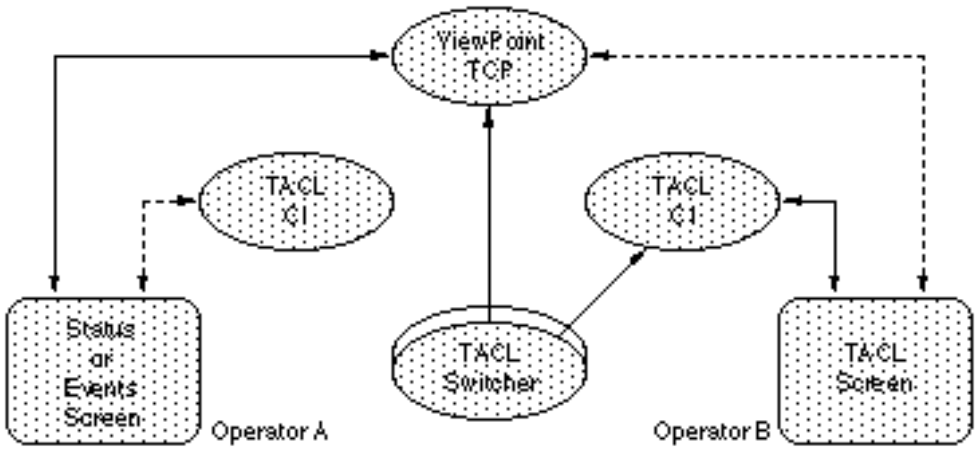


When you are done with TACL, you press one of the function keys for status, events, extras, or profile, which returns you to the ViewPoint environment. This causes TACL to send a message to the TACL switcher that tells it TACL is done and that it should reactivate your task in the ViewPoint process. Accordingly, the TACL switcher suspends your TACL interpreter and reactivates your task in the ViewPoint process, which then assumes control of your terminal.

When you press the Exit function key (SF16) from either environment, you cancel your terminal link to the ViewPoint process and give control to your TACL process. Your TACL is, at this point, outside the ViewPoint application, and you no longer have access to the ViewPoint function-key assignments for status and events.

Figure 5-5 illustrates the case of two network operators, identified as Operator A and Operator B. (It is possible to have more than two operators.) Operator A is using a task in the ViewPoint TCP for status or event screens. Operator B is using TACL. These active operations are indicated by bold line connections in the figure.

Figure 5-5. TACL Processes for Multiple Operators



TACL switcher processes are created dynamically, as needed, within a server class. Therefore, as illustrated by Figure 5-5, one TACL switcher (of the server class) is running; it is associated with the active command interpreter of Operator B.

Assume that at some later time, Operator A presses the TACL function key. This causes a message to the TACL switcher class, which creates a switcher for A; this switcher activates the TACL command interpreter for A.

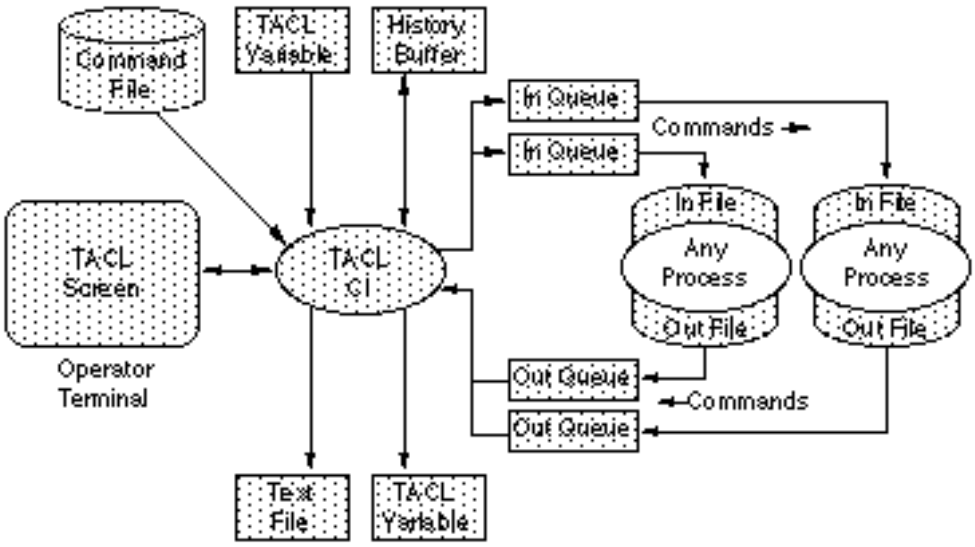
Conversely, Operator B presses one of the function keys for a ViewPoint screen, causing a message to the TACL switcher for B that reactivates the B task in the ViewPoint process. Then, the TACL switcher for B, no longer needed, terminates after some time interval previously set (the default is 10 minutes).

After these two switches, terminal interaction then proceeds as indicated by the dashed lines, rather than the bold lines.

Figure 5-6 illustrates the environment when you are using TACL with processes started by the Define Process feature. (The ViewPoint process is not shown.) Your TACL command interpreter accepts commands from your terminal keyboard, a file, or a variable. All commands are entered sequentially in your TACL history buffer.

Each command, whether simple or composite, is held in a TACL input queue while it is being executed. The command can result in commands sent to several processes, and each command is received by each targeted process in the file of that process. Command responses are returned from the out file of each process to a TACL out queue. Output can be saved in a text file or a TACL variable.

Figure 5-6. TACL Environment



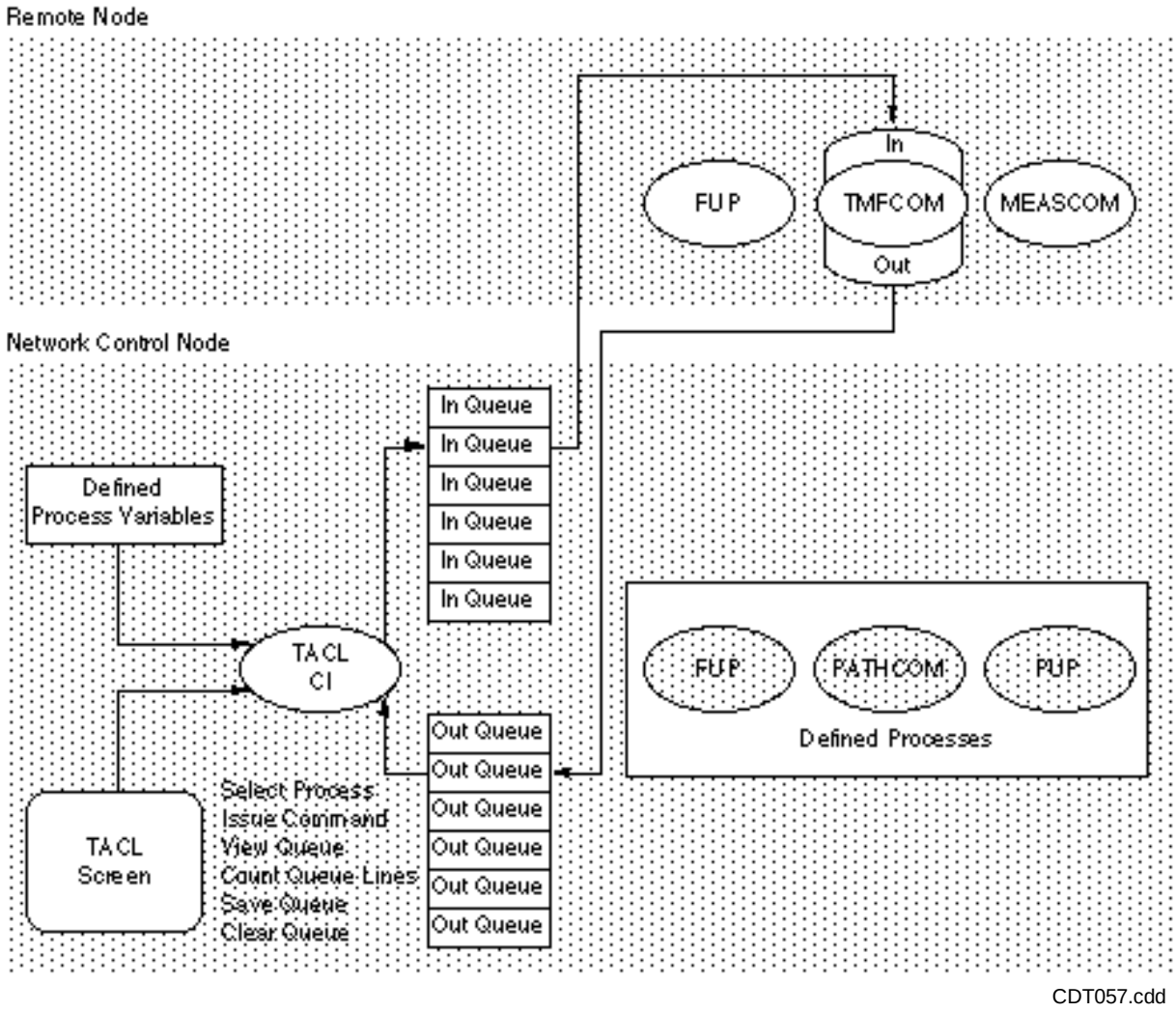
You can page through the display of commands and responses using the PREV PAGE (or PG UP) and NEXT PAGE (or PG DN) keys on your terminal (limited to the extent provided by your terminal's internal storage; the storage is cleared when you leave the TACL environment).

Define Process

Figure 5-7 shows the concept of the Define Process capability of the ViewPoint application. When you access TACL from inside the ViewPoint application, you have a set of Define Process variables that you can use from the TACL screen. (These can also be used in user-written application programs.)

Basically, the Define Process capability allows you to define a number (up to 100) of background processes. You can easily switch between processes without stopping and restarting every time you enter or exit a process. Your context is preserved when switching among the processes. You can define only conversational-mode (not block-mode) processes. Figure 5-7 shows six such background processes existing for one operator. Three are in the operator node (the NCN), and three are in some other remote node. Note that there is one "In Queue" and one "Out Queue" for each defined process.

Figure 5-7. Operator Access to Background Defined Processes



You define background processes with the DP command and undefine them with the UNDP command. Choose your own symbolic name for each process you define. Then, to pass a command to a defined process, you simply enter the symbolic name of the process followed by the command. Or, as usual for Compaq command interpreters, you can enter only the symbolic name (and omit the command) to initiate passthrough mode, in which the defined process issues its own prompt. In that case, you communicate directly with that process through the in and out queues.

You can specify various options when you define a process: the capability to have TACL interpret the function keys and intercept commands, and the ability to invoke a specific macro each time the process is invoked.

As indicated in Figure 5-7, Define Process commands that relate to the queues include commands that view the in or the out queues, obtain a count of the number of lines in the queues, save the contents of the queues in a file or variable, and clear the queues.

Refer to Section 4, Process Definition Commands for complete descriptions of the Define Process commands.

Status Presentation

The next five figures illustrate the concepts of status reporting and presentation. The first, Figure 5-8, shows an overview of the scheme used to present network status. A single operator terminal is shown, along with the three status screens.

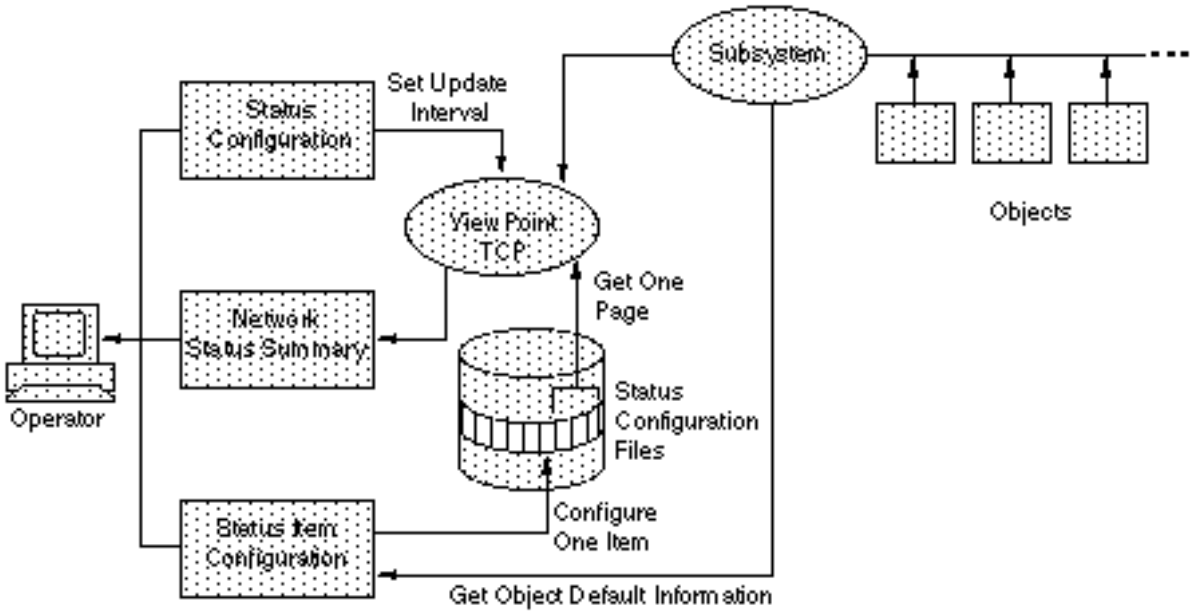
The Network Status Summary screen displays 16 lines of status information, one item per line, as reported by the subsystems.

The Status Configuration screen allows you to specify an update interval for updating the information displayed by the Network Status Summary screen. This value (in seconds) is retained and used by the ViewPoint TCP in requesting status from subsystems.

The Status Item Configuration screen enables you to create and modify status configuration files. You can choose to have several such files, providing you with a means to quickly select different kinds of status displays. You make the selection of a status configuration file on the Profile screen.

You use the Status Item Configuration screen to configure one item at a time. The order of configuration items in your file determines the order of items displayed on the Network Status Summary screen. Each set of 16 items (which can include label lines and blank lines) in your file constitutes a page of items.

Figure 5-8. Overview of Status Presentation



Configuring a status item primarily consists of specifying the kind of item (display type) and the system node from which to obtain the item. You must specify the particular object, and provide a description (up to 22 characters) that will mean something to you when the item is displayed on the Network Status Summary screen. Optional items for configuring are threshold highlighting, values for computing percentages, and a request to reverse the significance of status data. To reverse the significance of status data, for example, you could do the following: instead of using 0 to signify idle and 100 to signify a completely busy processor, you could use 0 to signify a completely busy processor (no capacity left) and 100 to signify idle (100% capacity remaining).

When you request the Network Status Summary screen (by pressing function key F2), the ViewPoint TCP retrieves one page of status-item configuration data from the configuration file and uses this data to get status information from the subsystems regarding each of the 16 objects. The ViewPoint process then displays the received status information on the screen. At periodic intervals (as specified on the Status Configuration screen), a new set of 16 update requests goes out from the ViewPoint process to get new status information from the subsystems for the page currently displayed.

To display other pages of status items, you can use the PREV PAGE (or PG UP) and the NEXT PAGE (or PG DN) keys, or you can specify a page number on the option line (line 24) and press F2 to go directly to that page. These keys fetch a different page of configuration data from the configuration file, resulting in status being requested for 16 different items.

You can always press F2 (Status) to force an immediate update of the screen. Also, you can use a Freeze key (F8) to inhibit automatic updates until you resume with the Thaw key (SF8).

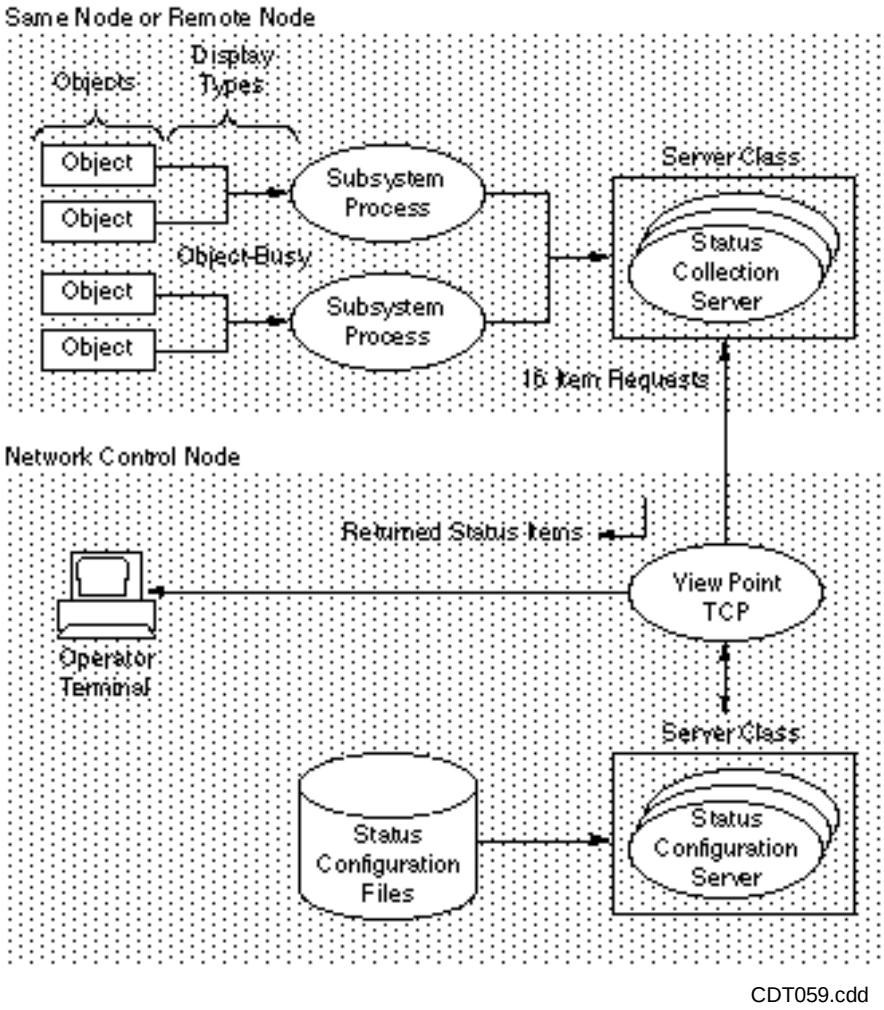
When you request the Primary Events screen or the Alternate Events screen from the Network Status Summary screen and then return, your screen context is preserved. That is, you return to the page from which you left, any selected status items are still marked, and the frozen or thawed state is the same. Screen context is not preserved when you move from page to page of the status display; in that case, only the frozen or thawed state is preserved.

Operation of Status Components

Figure 5-9 illustrates the main components of the status reporting function. The components shown in the upper half can be in a network node other than the NCN.

An object can typically be a CPU, a disk drive, a terminal, or a communications line. The display type can be an object-busy value (that is, the percentage busy during the configured sampling interval), or it can be a count of objects in a group that are currently started or running. It can also be some other type of value, such as transactions per second.

Figure 5-9. Main Components of Status Reporting Function



The subsystem process generically represents the source of status information. This can be a process for command processing (such as COMMGR for SNAX), a monitor process (such as PATHMON for Pathway), or simply a collection of callable procedures, as in the case of Measure.

Status collection servers are server classes—one or more processes sharing the work load but having a single server class name. The servers collect the requested status information and forward the information to the ViewPoint TCP.

The sequence of operations begins when you press the Status function key (F2). To satisfy this first request for status information, your task in the ViewPoint TCP obtains configuration data from the current status configuration file (through the status configuration server) for the first 16 items in the file. Your task formulates 16 individual status-item requests and sends them to the appropriate subsystems through the status collection servers in each network node.

When your task in the ViewPoint TCP has received the information for all 16 items, it performs the required percentage calculations (compensating for any failed subsystems that do not report status) and displays the results on the Network Status Summary screen on your terminal.

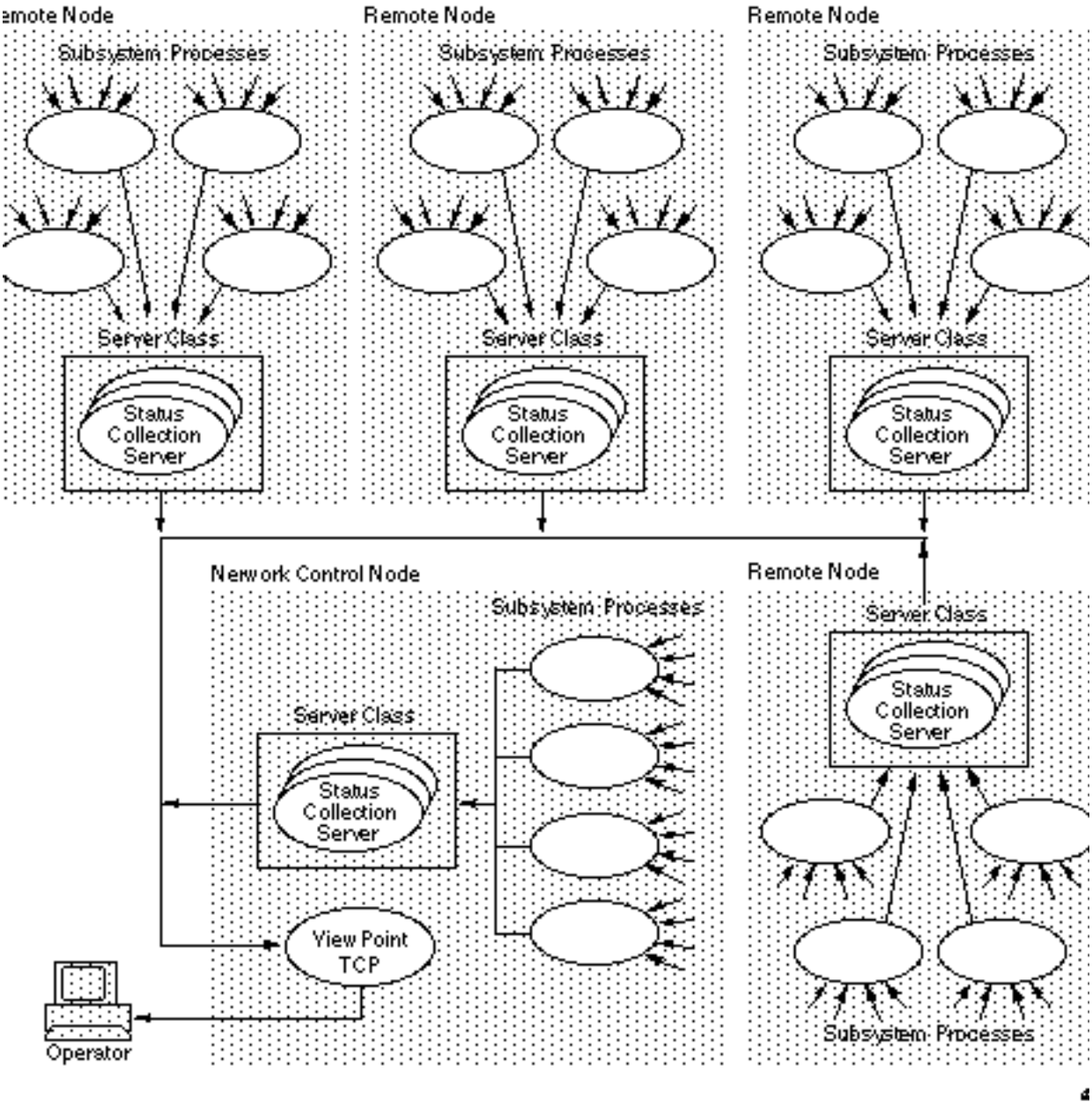
When you request another page of status information, your task in the ViewPoint TCP specifies this request to the status configuration server. In response, the status configuration server obtains the configuration data for that set of 16 items from your current status configuration file and returns the data to the ViewPoint TCP. Thus, status is requested, as before, for these 16 new status items.

Status Reporting in Networks

Figure 5-10 depicts status collection extended to a network—in this case consisting of one NCN and four remote nodes. Each node is assumed to have four subsystem processes under the control of one status collection server in each node. Input arrows to the subsystem processes represent status input from various objects.

As shown, all five status collection servers supply status information to the ViewPoint TCP in the NCN when commanded to do so. The network ViewPoint TCP then displays the received status information on the Network Status Summary screen.

Figure 5-10. Status Reporting in a Network

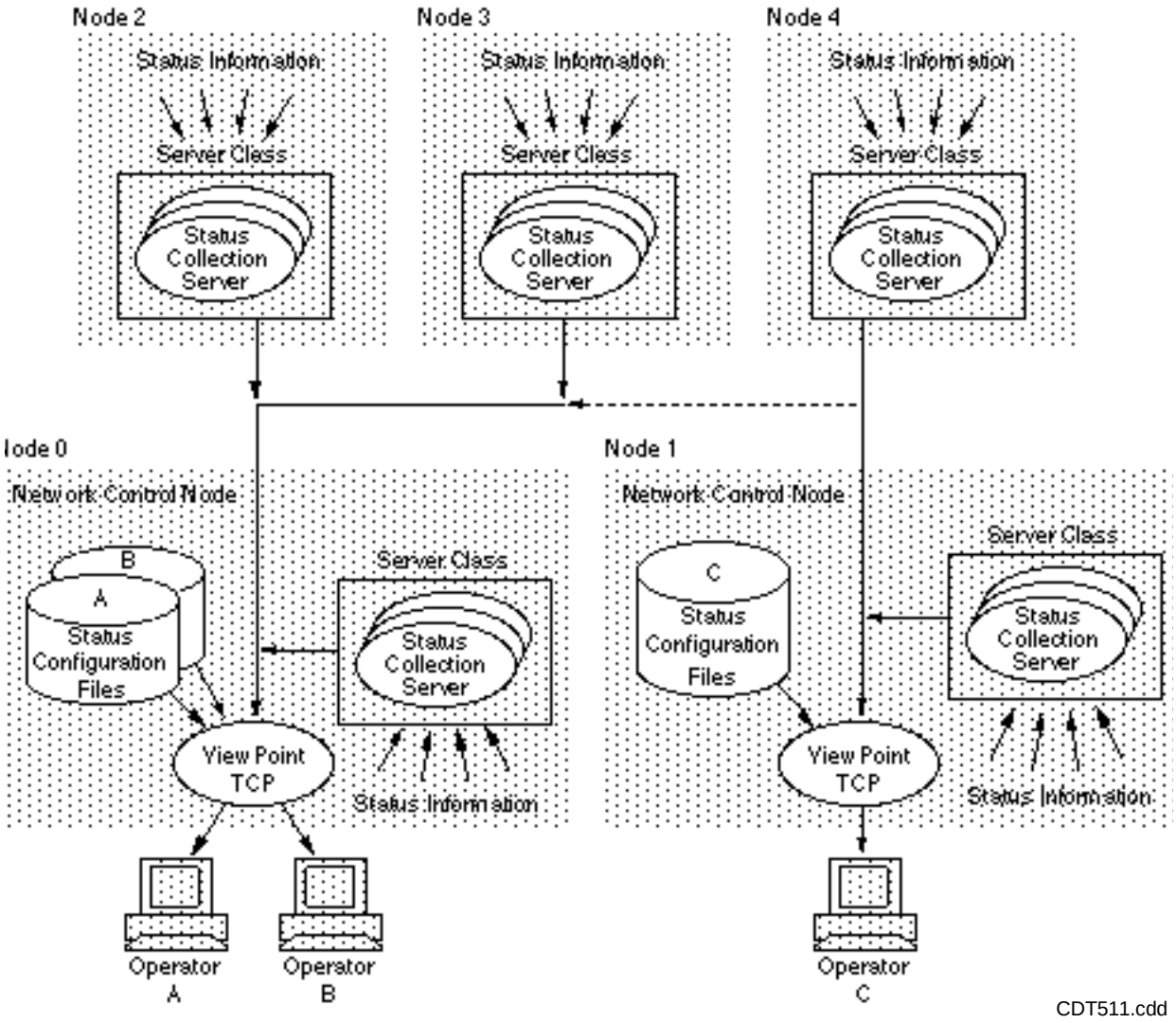


If more than one operator is using your network, each can receive different status reports. Figure 5-10 shows status information is supplied to the status servers from subsystem processes.

Figure 5-11 illustrates the case of three network operators, showing the same network as in Figure 5-10. In this case, however, one of the remote nodes is now set up as a second

NCN. Thus the term remote is not used, and the nodes are simply numbered 0 through 4. For simplicity, the subsystem processes have been omitted.

Figure 5-11. Reporting Status to Multiple Operators



Node 1 (a network control node) obtains status information only from itself and from node 4.

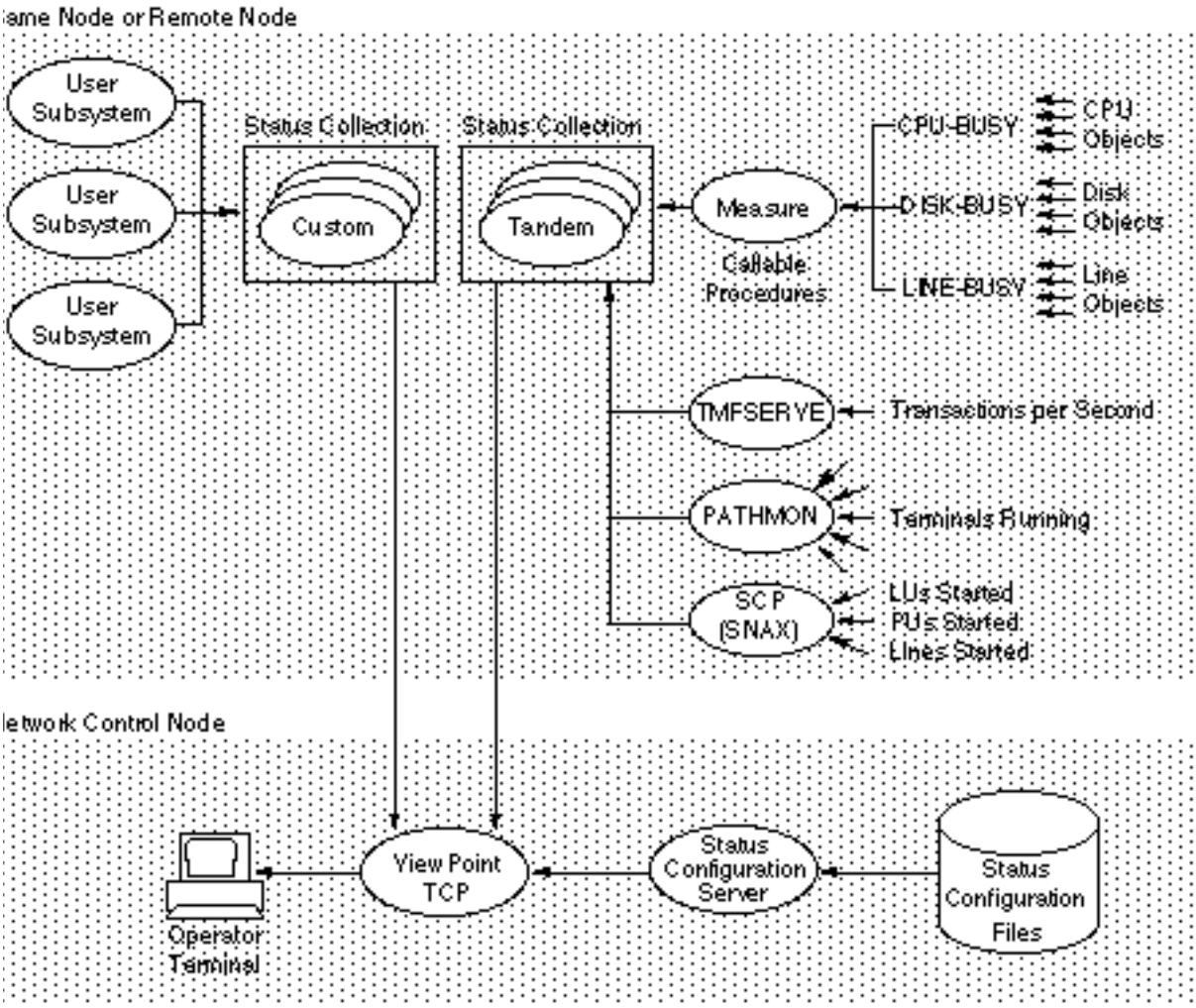
Node 0 (the other NCN) obtains status information from itself, from nodes 2 and 3, and (note the dashed-line connection) from node 4. Getting status from node 4 is possible, even though the status collection server in node 4 belongs to the ViewPoint TCP in node 1.

Operator A and Operator B are logged on at Node 0, and Operator C is logged on at Node 1. Note that each operator has his or her own set of status configuration files—(labeled A, B, and C). Thus each operator can select independent status displays.

Example Configuration of Status Servers

Figure 5-12 shows an example of status information being collected from a single node that could be the NCN or another node. The node has two status collection server classes. One is provided by Compaq, and the other is a user-written custom server.

Figure 5-12. Example of Status Reporting Configuration



CDT512.cdd

The Compaq status collection server obtains status information from the Compaq subsystems Measure and SNAX and the subsystem processes TMFSERVE and PATHMON; the Compaq status collection server then forwards the information to the ViewPoint TCP.

The types of information obtainable from Measure are CPU-BUSY and AVERAGE-CPU-BUSY percentage values from CPU objects in the node, DISK-BUSY percentage values from disk objects in the node, and LINE-BUSY percentage values for communications lines connected to the node.

TMFSERVE provides a count of transactions per second, measured at the time of each status request from the NCN. PATHMON provides a count of Pathway terminals currently running on a particular Pathway system. The SCP process for SNAX provides a count of SNAX LUs started, PUs started, and lines started.

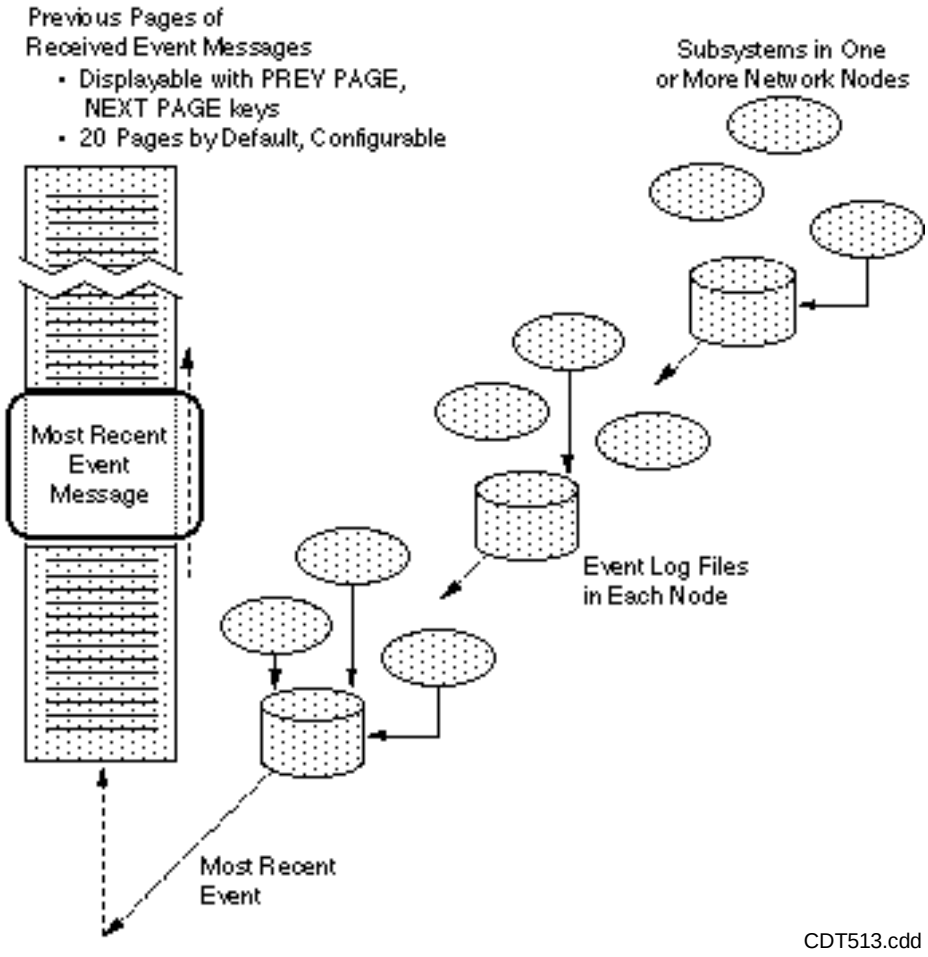
The user-written status collection server obtains status information from the user application subsystems and forwards the information to the ViewPoint TCP in the NCN.

Event Presentation

The remainder of this section describes the concepts involved in collecting and presenting network events.

Figure 5-13 illustrates the general concept of displaying events in current-events mode. (Historical mode is considered later.) Sources of event messages are subsystems that have message-reporting capability. All such event messages are recorded in an event log in each network node. In addition, each new message is forwarded to the NCN for display on your terminal.

Figure 5-13. Operator View of Event Presentation



On your terminal, one line of text is displayed for each event in the order received. The most recent event is inserted at the bottom of the page, and older ones roll off the top of the page. In this mode, the page that displays the 16 most recent event messages is called the END page. You can review earlier pages by using the PREV PAGE (or PG UP) and the NEXT PAGE (or PG DN) keys, or by entering a page number on the option line and pressing F3.

Each of these earlier pages is 16 lines in length. By default, a total of 20 pages can be scanned this way, as determined by the size of an event cache that you can configure to some other number of pages. (The maximum is 750 pages.) Older event messages roll off the end of the cache as new ones arrive; however, you can still review them by accessing the copy retained in the event logs using historical mode. The cache exists to provide you with quick access to some convenient number of recent messages.

The END page continues to be updated (invisibly if you are viewing some other earlier page) as event messages are received. When you do view the END page, you can stop the updates and consequent scrolling of the screen by using the Freeze function key (F8)

and then resume updates by using the Thaw function key (SF8). You can specify a delay time so that updates do not occur too rapidly for you to read comfortably.

When you request a particular page on the Primary or Alternate Events screen, the content of the displayed page is relative to the current END page. As new events arrive, the END page changes, and the set of events displayed for a particular page number changes accordingly.

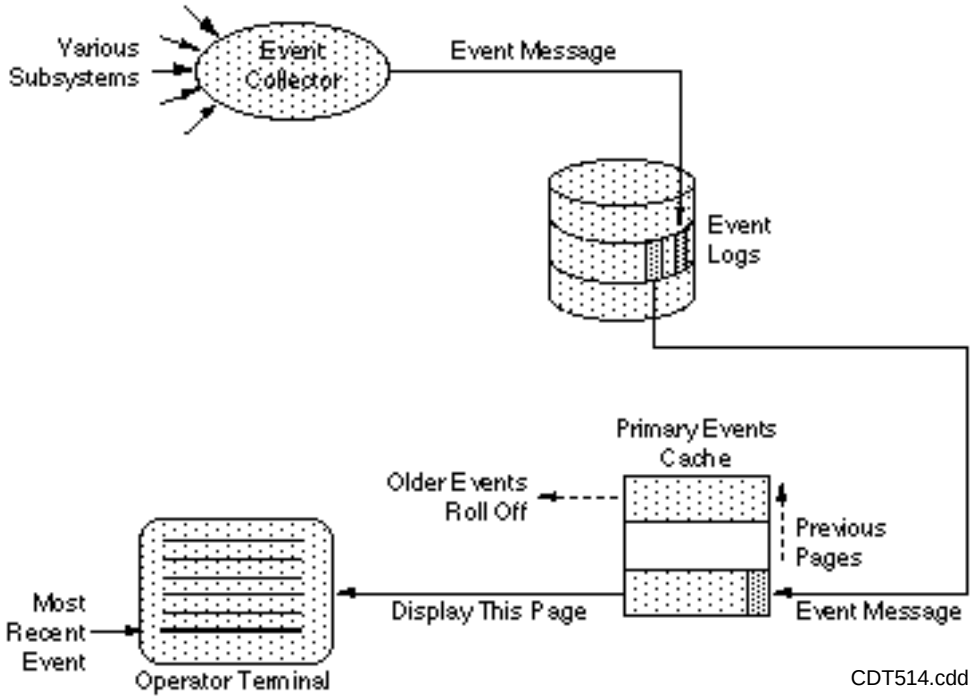
When you request the Network Status Summary screen or one of the other Events screens from either the Alternate Events screen or Primary Events screen and then return, your screen context is preserved. That is, you return to the page from which you left, any selected event messages are still marked, and the frozen or thawed state is the same. Screen context is not preserved when you move from page to page of the events display; in that case, only the frozen or thawed state is preserved.

Current and Historical Modes

Figure 5-14 shows the basic mechanism for the current-events mode. The primary-events cache is always used for current events, and there is only one primary-events cache in an NCN (unless there is more than one ViewPoint application running).

The primary-events cache belongs to ViewPoint and is shared with other operators, if any. The event collector and event log are part of Event Management Service (EMS).

Figure 5-14. Collecting and Displaying Current Event Messages



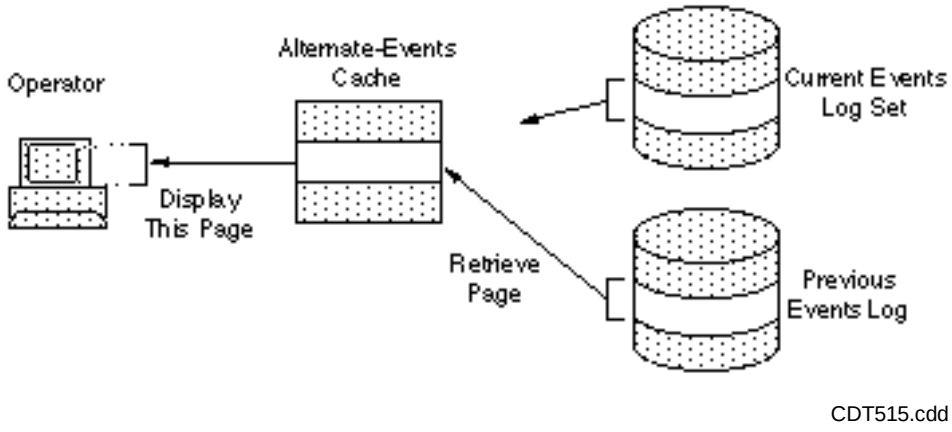
In this basic scheme, subsystems report events in the form of event messages sent to the event collector. The event collector records these event messages in an event log in the

order received. The Event Management Service detects that new messages are in the log and accordingly forwards the next message in order to the ViewPoint primary-events cache. This forwarding of messages causes the oldest message to roll off the other end of the cache if the cache is full of event messages.

If you are displaying the END page of current events, the newest message appears as the 16th message on your screen. Meanwhile, a new request is returned to the Event Management Service to get whatever event message is next in order.

Figure 5-15 shows the basic mechanism for historical mode. This mode can be used only with your alternate-events cache. Unlike the primary-events cache, the alternate-events cache is not shared; it belongs to you as long as you are running ViewPoint. (You can also use this alternate-events cache for other special views of events, as described later in this section.)

Figure 5-15. Retrieving and Displaying Events in Historical Mode



You use the historical mode to examine past events for problem analysis and diagnostic purposes. It provides access to events older than those in the primary-event cache (which is only 20 pages or so, depending on your configured size). As indicated in Figure 5-15, event messages can be retrieved from the current-events log file set or from one of several other event logs (such as archived logs or logs belonging to a collector other than $0 on your local node). You can specify other event logs in the field called Collector or Log File on the Event Configuration screen for alternate events.

To use historical mode for the current log set, specify a starting date and time in the field “After:” in the Event Configuration screen. If the date is earlier than the dates recorded in the file that EMS is currently using for new events, EMS locates the appropriate log file elsewhere in the current log set. Then, when it has found the starting point in the required file, EMS begins to retrieve error messages from that point, one by one, and delivers them to the ViewPoint application for copying into your alternate-events cache. This overwrites any previous contents in that cache.

If you specify another collector or log file in the alternate event configuration, EMS similarly retrieves events from the specified date and time in the log.

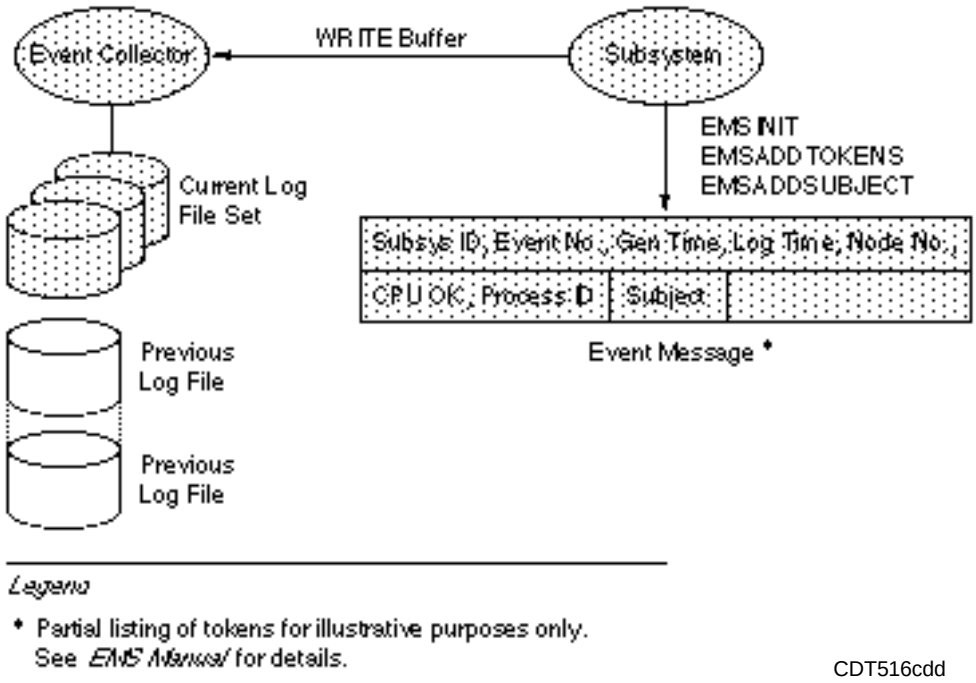
At this time, you can view this first page of historical events on your screen, if you select alternate events (function key SF3). If the cache is not yet full, the display continues to be updated with incoming messages until the cache is full. No further pages can be retrieved. This is indicated by the word END displayed at the top right corner of the Events screen. You can view only pages currently in the cache. To retrieve succeeding or preceding pages from the cache, you would have to set a new starting time on the Event Configuration screen and start over.

Logging of Events

Figure 5-16 begins a sequence of diagrams that explore the event reporting scheme step by step, concluding with a composite view at the end of the section.

The first step, as illustrated in Figure 5-16, is the logging of event messages in a log file. This step begins with the originating subsystem that detected the event. The subsystem writes all pertinent information about the event into an event-message buffer. This includes the subsystem ID and event number.

Figure 5-16. Logging of Events



Much of the information in the event-message buffer is in the form of tokens, consisting of a token code and a token value. The tokens indicate the following:

- Time the event occurred
- Network node number
- Process identification (PID)

- Specific subjects affected
- Event is a critical event (emphasis token) or an action event (action token)

The kinds and number of tokens included in the buffer can vary from subsystem to subsystem.

Once the buffer is complete, it is sent to the node's event collector process. The event collector inserts its own token representing the log time and records all this information in its current log file.

The current log is a set of files linked together so as to be addressable as a single log. Typically, an operator has several archived log files in addition to the current log. An archived log file is a file that was previously part of the current log set. Succeeding discussions assume that only the current log set is considered unless an archived file is specifically mentioned.

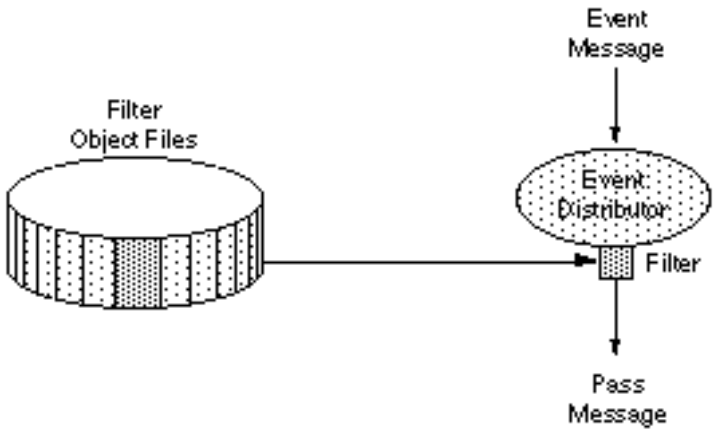
Event Distribution

Once event messages have been collected in an event log, the next step is to distribute these messages to the network operators at one or more NCNs. Event distributors are processes that perform this distribution work.

To allow selectable distribution of event messages, event distributors allow installation of a filter. Filters consist of processing logic that permits distribution (passing) of certain kinds of event messages, while inhibiting the distribution of those messages that do not meet criteria specified in the filter.

Figure 5-17 illustrates the concepts of loading a filter into an event distributor process and using the filter.

Figure 5-17. Filtering Event Messages



Event Filters

Each filter is contained in a separate filter object file. For primary-events displays, these files are kept in the ViewPoint program subvolume. One filter is loaded at startup time, and it cannot be changed until the ViewPoint TCP is stopped and restarted. This filter is common to all operators logged on at a given NCN.

For alternate-events displays, you, as one of several operators using the same ViewPoint system, can compile your own filters, store them in your own subvolume, and load any one filter at any time. You can also pass parameters from the Alternate Event Configuration screen to the currently loaded filter for dynamic filter definition.

When an event distributor receives an event message, it compares the token codes and other information contained in the message with token codes and other information specified in the currently loaded filter. If there is a match, the message is passed on to the next process in the event-handling chain; otherwise, it is not passed on (although it continues to exist in the event log).

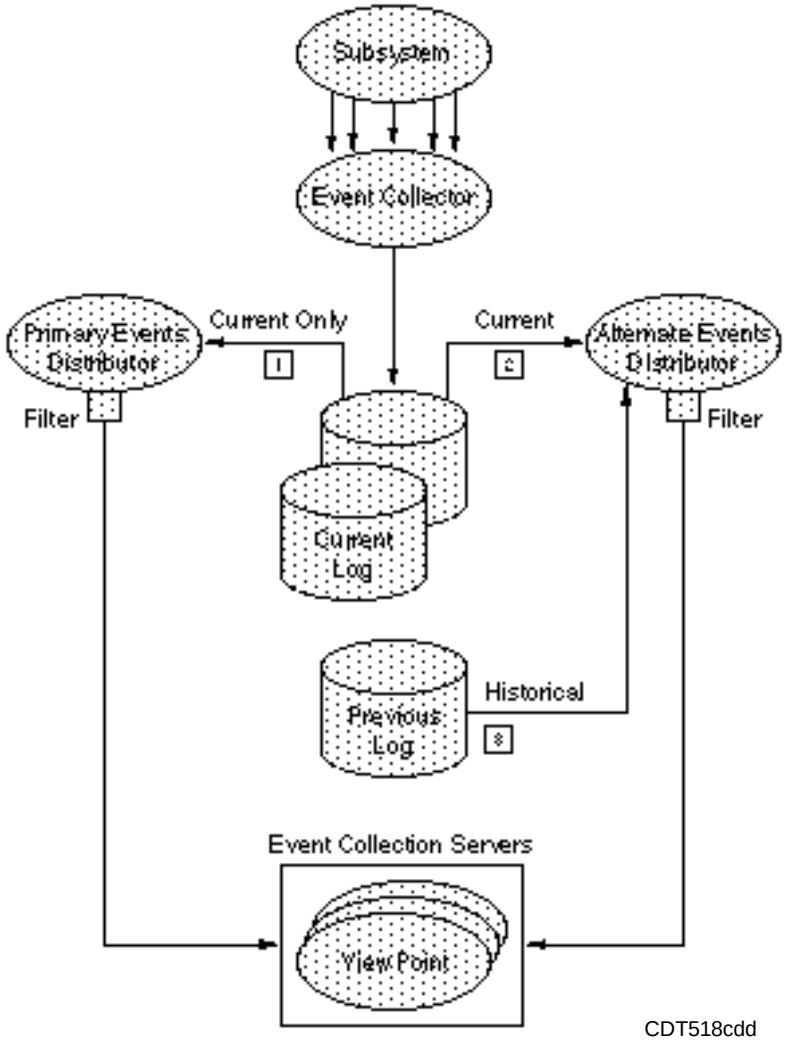
Loading a filter consists of copying it from its disk file to the data space of an event distributor process. This makes the filter available to the filter comparison logic of the process.

Filters are stored in filter files by the event-management filter compiler, one filter per file. This is described in the *EMS Manual*.

Event Distributors

Figure 5-18 illustrates the three ways in which event distributors access the collector's current log set. These three ways are numbered 1, 2, and 3 in the figure. Note that there are two event distributors: primary-events distributor and alternate-events distributor. Both are shown to be reading from the current log file set, and both send their filtered messages to an event collection server.

Figure 5-18. Reading of Log Files by Event Distributors



The ViewPoint application defines one primary-events distributor in an NCN (assuming only one instance of ViewPoint is running). The alternate-event distributor, however, belongs to the operator, and there can be one associated with each alternate-events display. The operator can configure the alternate-event distributor to read from a specific log file in any selected node.

The following three paragraphs describe each of the three cases shown in Figure 5-18.

1. The primary-events distributor has the single purpose of distributing current events for primary-events displays for the network operators logged on at this node. Its filter is not modifiable by individual operators; the filter is installed in the initial configuration setup.

2. If you specify current-events mode for your alternate-event display (by not giving a date and time), your alternate-event distributor reads current events from the current

log—just like the primary-events distributor. However, in this case, you can specify your own filter for reading only selected events (note the filter shown attached to the alternate-event distributor). That is something you cannot do with the primary-events display.

3. If you specify historical mode for your alternate-event display (by specifying a date and time), your alternate-event distributor reads historical events from either the current log set or (as shown in Figure 5-18) from an archived log file. As in the preceding case, you can specify your own filter for reading selected events from that file.

In either of cases 2 or 3 above, if some other operators choose to read from the same log file that you are accessing, those operators have their own alternate-event distributors and their own filters.

Event Distribution in Networks

Up to this point, only a single system has been considered. In the case of the multiple nodes of a network, event messages must be distributed from originating nodes to one or more NCNs. There are several ways to do this kind of distribution. The more common method uses a special kind of event distributor, called a forwarding distributor. Another method specifies multiple collectors on the remote nodes as the source of events. A third method specifies only the event collector on a remote node. For EMS distributors running on C-series systems, ViewPoint gives full support.

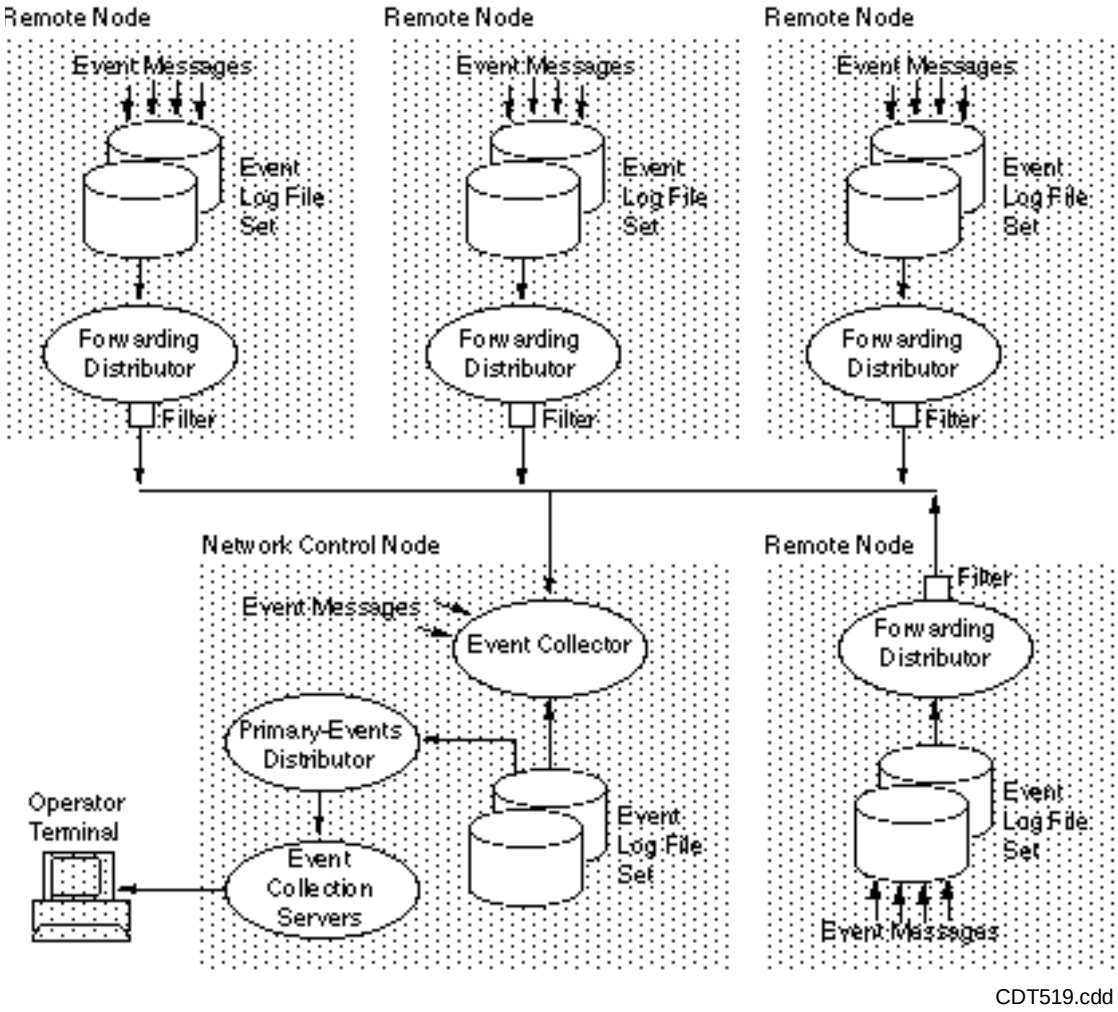
When events are forwarded, there is one forwarding distributor on each node for each NCN. That is, if two NCNs are in a network, separate forwarding distributors are required to send event messages to the two NCNs. When multiple collectors are specified, then the events from each remote collector are merged by the ViewPoint event collection server on each receiving NCN. When a single remote collector is used, ViewPoint starts a remote distributor on that node to distribute events to the local NCN—unless the configuration specifically assigns a distributor to a particular node. Each of these methods is discussed in more detail in the following subsections.

Forwarding Distributors

Figure 5-19 is a simplified example of the collection and distribution of primary events in a network. It focuses on the role of the forwarding event distributors. (For simplicity, the subsystem processes and the event collector are not shown in this figure.) Previously shown in Figure 5-18, event messages from the subsystems are collected and recorded in the current event log file by the event collector.

The diagram in Figure 5-19 shows one NCN and four remote nodes. Each node, including the NCN, records event messages in its current log file. Each forwarding event distributor in the four remote nodes distributes event messages from its associated current log to the event collector located in the NCN. The NCN event collector merges all these incoming event messages from other nodes with messages generated locally and records them in its own current log file.

Figure 5-19. Event Collection and Distribution in a Network



From the central events log, event messages are read out by the primary-events distributor, and distributed to you and any other operators logged on at the same NCN through the ViewPoint event collection server.

If more than one operator is using the same ViewPoint system, all operators receive the same primary-events messages. Figure 5-20 illustrates the case of three network operators, showing the same network as in Figure 5-19. One of the remote nodes is now set up as a second NCN, so the nodes are numbered 0 through 4.

Figure 5-20. Distributing Events to Multiple Operators

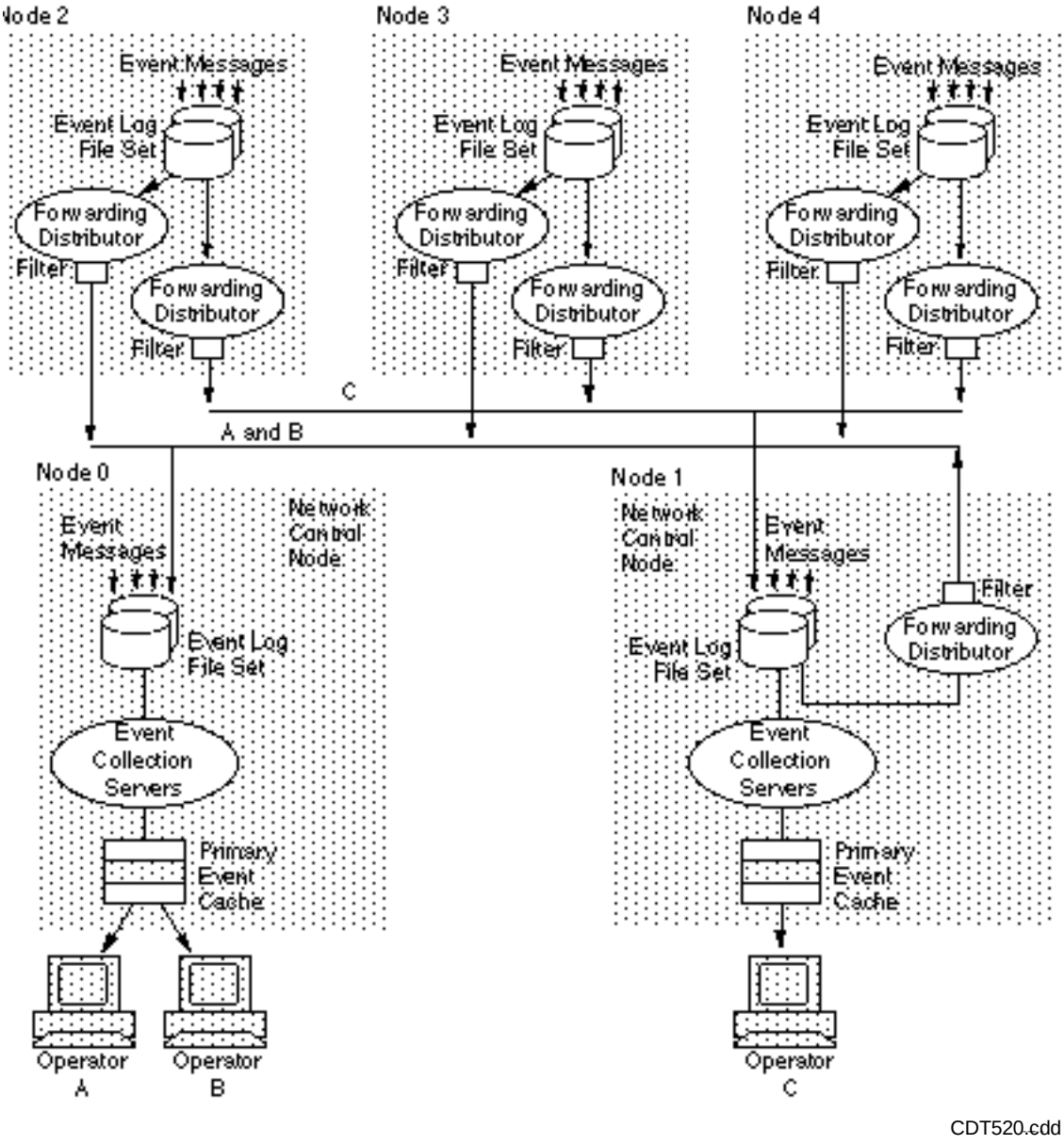


Figure 5-20 (on the previous page) shows only the key processes in this network configuration; other processes involved are shown in figures later in this section. The NCN event log in node 0 receives event messages from all five nodes—from its own local system and from four forwarding distributors in the other four nodes. Note the connecting lines labeled A and B. The NCN event log in node 1, however, receives event messages from only four nodes, with node 0 being omitted. Note the connecting line labeled C.

Operator A and Operator B are logged on at Node 0, and Operator C is logged on at Node 1. Note that Operator A and Operator B read from the same events cache, which gets current event messages from the NCN event log through the event collection server.

Multiple Collectors

Up to five remote collectors can be specified as the source of events for each of the event screens (primary, alternate, or last events). The events from these collectors are sent directly to the event collection servers on the node where they are to be displayed.

The operator at an Alternate Events screen can specify from 1 to 5 collector names as the source of events displayed on his or her screen.

The operator at a Primary Events or Last Events screen has no control over the collectors. For these screens, multiple collector names must be specified at configuration time. Two PATHCOM assigns determine the collector names for the Primary or Last Events displays. (Refer to the description of the PRIMARY-COLLECTOR and LAST-EVENT-COLLECTOR assigns in Appendix A, “Server Assigns and Parameters.”)

When events are sent directly from named collectors at remote nodes, the events are not filtered. Sending unfiltered events can increase network traffic. As a general rule, it is more efficient to send events through forwarding distributors that use filters to reduce the number and size of events sent over the network.

Remote Distributors

If only one collector is specified and that collector is at a remote node, ViewPoint starts a remote consumer distributor at that node. In this case, filters can be associated with the distributor so that filtered events are sent over the network.

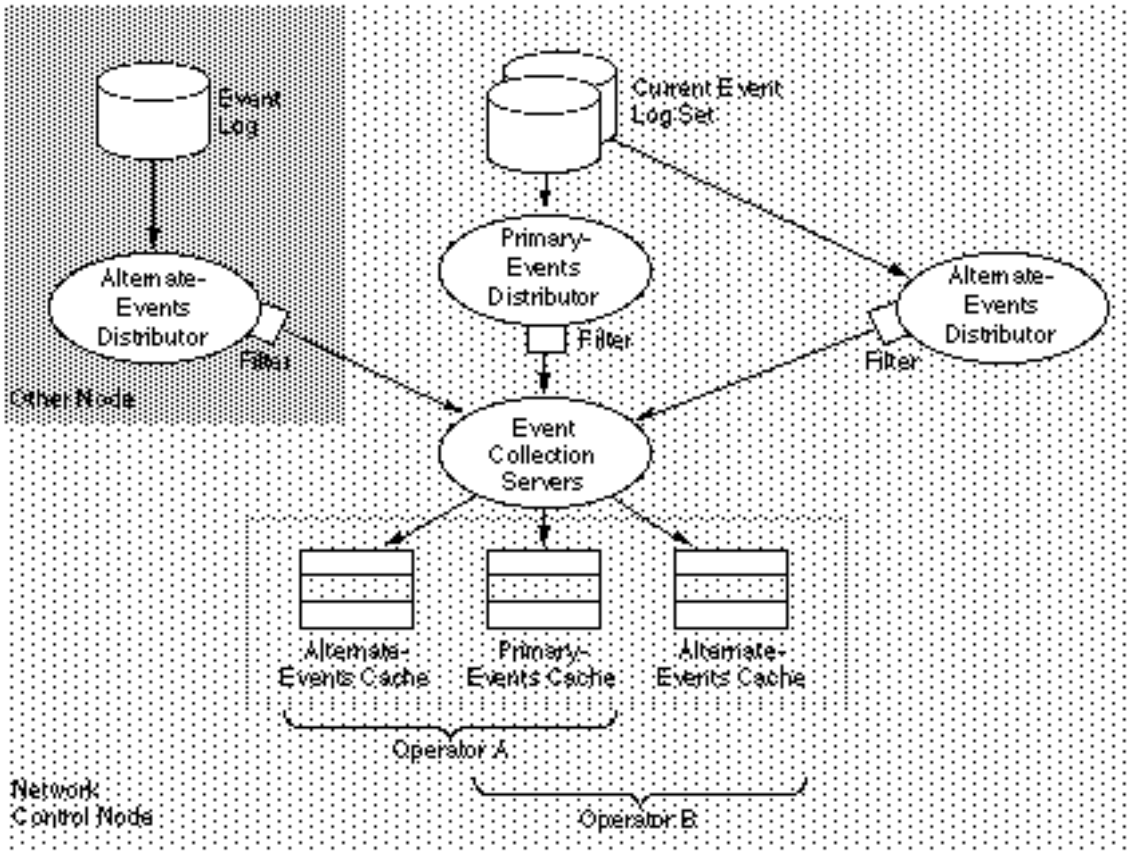
At configuration time, the person managing the system or ViewPoint application can include a DISTRIBUTOR assign in the Pathway configuration for ViewPoint to alter the distributor program file. When this assign is used, ViewPoint always starts the distributor using the specified program file, which can be either local or remote.

Event Display Operations

Figure 5-21 shows how primary-events and alternate-events displays are managed for multiple operators at the same NCN. Operator A and Operator B are served by separate tasks in the event collection server, which manages each operator’s primary-events and alternate-events caches. (Figure 5-21 and succeeding figures assume a single network control node.)

Note that alternate events are obtained directly from an alternate-events distributor, and that this distributor is located on the remote node, if it is reading events from a remote log. Each operator views only those events they want by having their own filter installed in their own alternate-events distributor.

Figure 5-21. Distribution of Primary and Alternate Events



CDT521.cdd

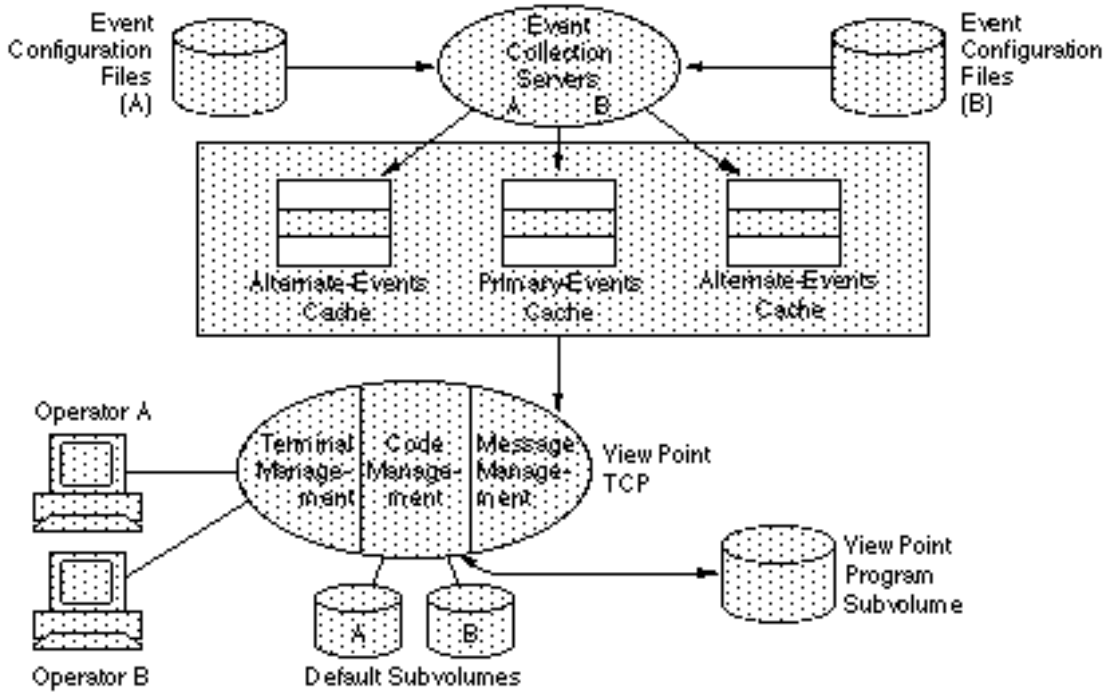
The selected events are passed on through the filter to their own individual tasks in the event collection server. In turn, the task in the event collection server inserts these event messages into each alternate-events cache stored in its memory.

Primary events, however, consist of incoming current events from possibly several nodes and are shared by both operators. When each operator views the Primary Events screen, that operator is observing current events contained in the primary-events cache.

For primary and alternate events, the event messages from remote nodes are passed through the filter associated with the forwarding distributor of the node. For simplicity, the forwarding distributor is omitted in Figure 5-21; refer to the two preceding figures for details.

Figure 5-22 is a continuation of the previous figure, concluding the sequence that began in Figure 5-16. This illustration shows elements for two operators. Each operator has the following: one set of configuration files, one alternate-events cache, one events-display process, one default subvolume, and one terminal.

Figure 5-22. Event Presentation Processes for Two Operators



System components (one each in any NCN) consist of the following: two event collection servers (one for last events, the other for all other events) and the ViewPoint TCP and its program subvolume. (There is also a ViewPoint data subvolume, not shown.)

The event collection servers store filtered event messages in the single primary-events cache and the alternate-events caches for each operator. These messages are then passed to the ViewPoint TCP for display.

The ViewPoint TCP is described in the overview earlier in this section. As stated there, ViewPoint TCP is the terminal control process of a Pathway application. The terminal-management function routes information to and from each operator. The code-management function manages the multithreaded execution of the application programs for each operator. The message-management function communicates with the remainder of the system.

The operator's default subvolume contains the operator's TACLCSTM, ZZVPSTAT, and ZZVPEVNT files, and a ZZVPCLIP file for saving copies of displayed screen information.

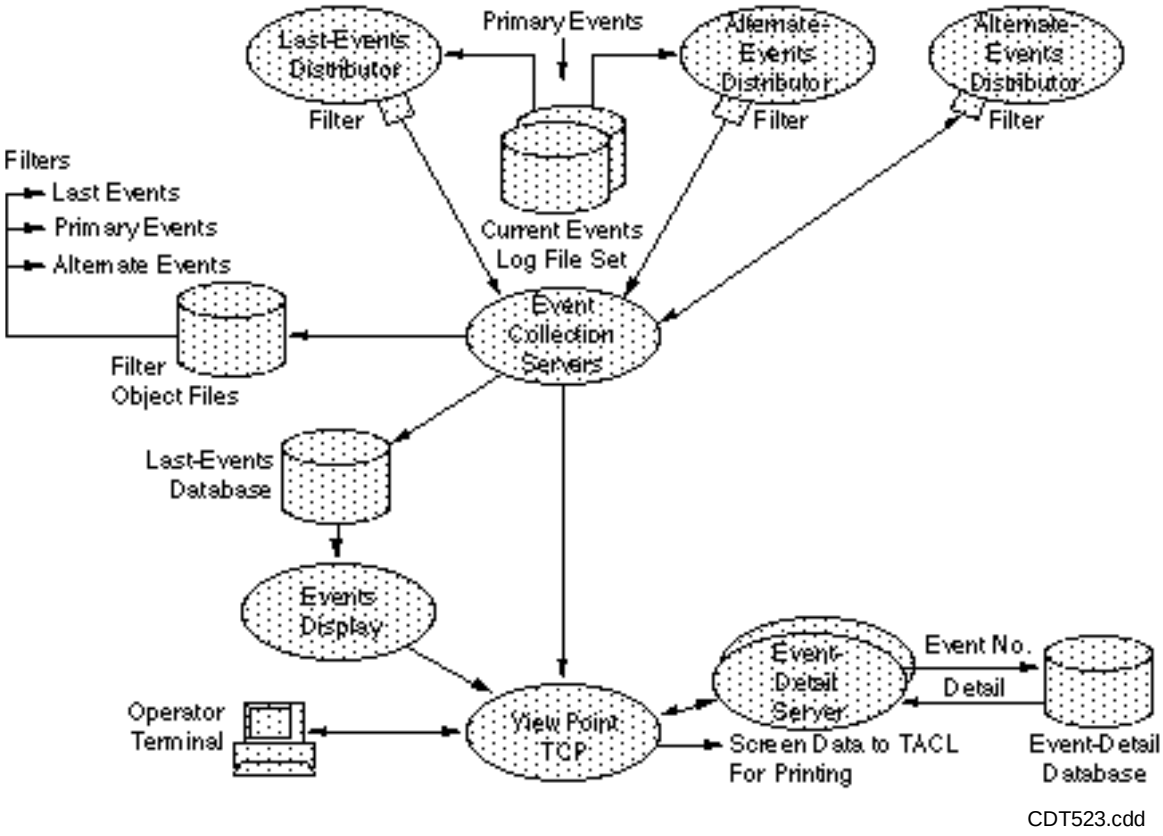
The ViewPoint program subvolume contains the ViewPoint application programs (SCREEN COBOL programs) and filters FLTRDFLT and FLTRLAST. The ViewPoint data subvolume contains the event-detail database and the event and status caches.

Other Event Operations

Figure 5-23 illustrates four additional features not covered in the preceding figures: display of last events for a subject, loading of filters, display of event detail, and printing of screen messages as a printed log. These features are separately discussed in the next four paragraphs.

The last-events distributor examines all event messages that come in to the NCN event log. All event messages that pass the existing last-events filter (which selects subjects of interest) are sent to the last-events collection server (a ViewPoint event collection server) for recording in the last-events database. The event messages in this database are organized on the basis of subjects. Thus, the events-display process (when you request the Last Events screen) can retrieve the most recent page of event messages for a given subject.

Figure 5-23. Other Event Presentation Features



You load an alternate-events filter (as shown earlier in Figure 5-17) by entering a filter file name on the Event Configuration screen and pressing the Configure function key. This causes the ViewPoint TCP to send a command directly to the specified alternate-events distributor process. The distributor reads the filter from the specified file and loads the filter into the data space of the distributor's filter in memory. (Filters for primary events and last events are loaded only when ViewPoint is started.)

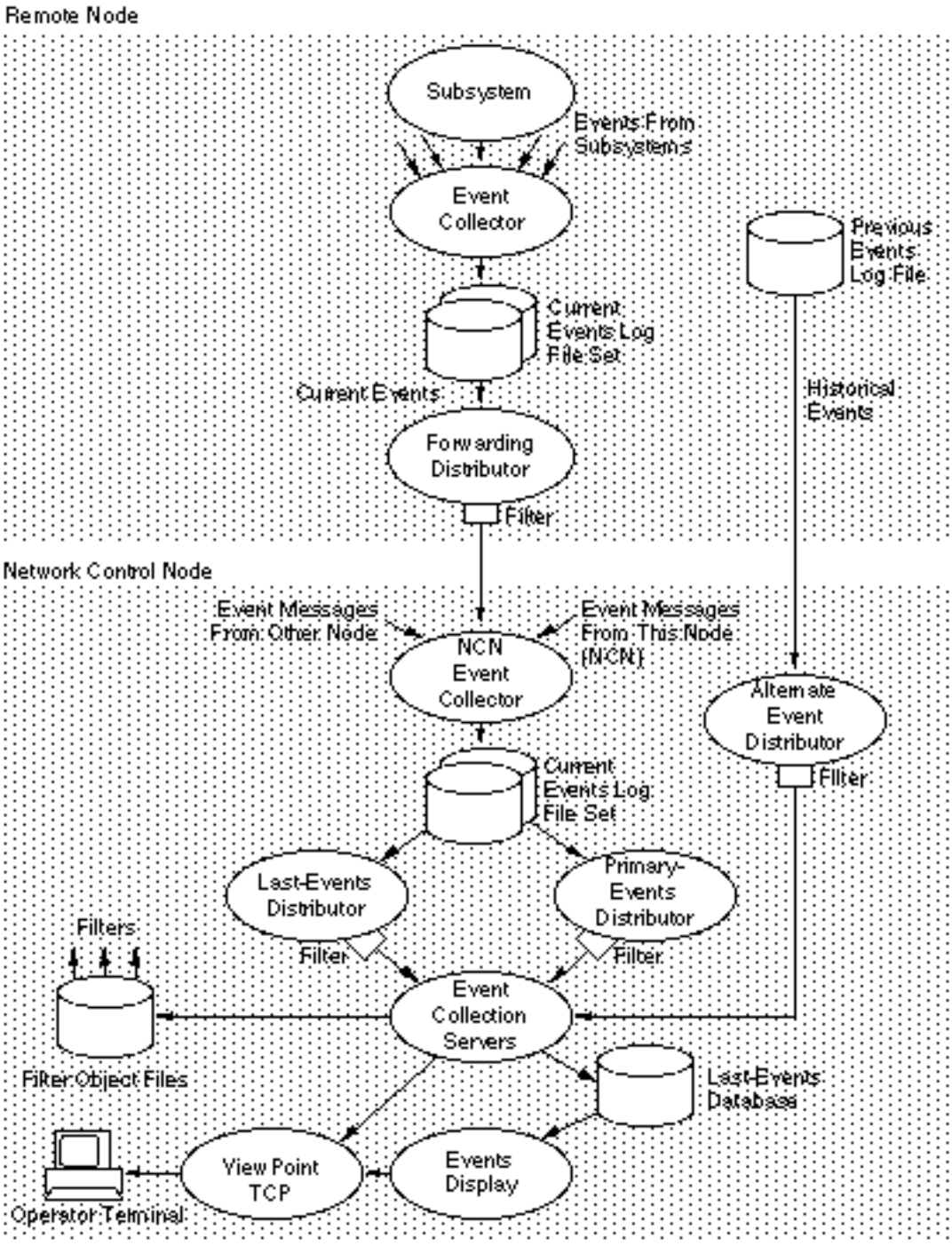
If a displayed message is truncated (because it is longer than 62 characters) or if you should want an expanded explanation of a given event message, you can request the additional information by going to the Event Detail screen. The ViewPoint TCP retrieves the information from the event-detail database through the event-detail server (a server class of one or more processes) and displays it on your screen. The expanded explanation, when present, includes probable cause, effect, and recommended action.

For printing and copying of screen information (when you press the Print and Clip function keys), the ViewPoint TCP communicates with TACL, which communicates the screen data externally to the Compaq *NonStop*™ Kernel operating system (not shown in Figure 5-23).

Summary of Event Operations

Figure 5-24 is a composite of the event-presentation architecture diagrams shown earlier (Figures 5-18, 5-21, 5-22, and 5-23). It is simplified to show one item of each kind: one NCN, one remote node, one subsystem generating events, one event collector, one forwarding distributor, one alternate-events distributor, and one network operator.

Figure 5-24. Event Presentation Architecture



The following six paragraphs summarize event operations:

- Logging a current event
  The subsystem that generated the event sends an event message to the node's event collector, which records the event message in the current log file.

- Transferring event messages to the NCN
  The forwarding distributor gets the current log file name from the event collector, reads newly logged event messages from the log, puts them through the distributor filter, and (if they pass) sends them to the NCN event collector. The NCN event collector records these messages in its current-events log—along with messages coming from other nodes—in the order received.

- Presenting the event
  The primary events distributor reads newly logged event messages from the NCN event log. If an event matches the filter that is currently loaded (for all operators), the event message is passed to your operator task in the event collection server. The event collection server puts the message into the primary-events cache stored in its memory. If you select the END page of the primary-events display, the page in the cache that contains the most recent events is sent to the ViewPoint TCP for display on your terminal. If you request an earlier page, that page from the cache is sent.

- Presenting the last events
  The last-events distributor also examines newly logged event messages. If these newly logged event messages pass, the currently installed filter (which specifies the desired set of subjects) passes these messages to an event collection server. Passing the messages to an event collection server puts the messages into the last-events database, where they are sorted and stored according to subject name.

  Thus, these records can be retrieved to show the most recent events for a given subject. Presenting last events for a subject begins when you request the Last Events screen, specifying a particular subject and pressing the Last Events function key. Your events-display process reads a full page of event messages (or less if the wrap-count for the specified subject is less than 16) from the last-events database and passes it to the ViewPoint TCP for display on your terminal.

- Loading an alternate event
  An alternate filter gets specified on the Alternate Event Configuration screen. When you press the Configure function key to accept this new configuration parameter, the filter is loaded from the specified filter object file.

- Presenting historical events
  The display of historical events begins when you request (on the Event Configuration screen for alternate events) a particular starting time (After field), plus possibly a node name and a collector name. This causes an alternate-events distributor process to be created for you, associated with the current (or specified) collector. (If you happen to have a distributor process elsewhere, it is stopped at this time.) Then, the After field information is forwarded through the event collection server to the new distributor. If the distributor does not have the file name for the file that corresponds to your specified date and time, it requests the collector for that name.

  Once the distributor has the log file name, the distributor can access the log file, beginning at the specified time, and it begins to retrieve event messages for your alternate-events display. These messages are sent to the event collection server,

which stores them in the alternate-events cache. A page of event messages is sent from the cache to the ViewPoint TCP for display on your terminal.

You can use function keys to switch among the primary-events, alternate-events, and last-events displays. The events-display process manages the selection. Screen context (page number, marked events, and frozen or thawed state) is preserved when you switch among these event screens.

Extending the Functionality of ViewPoint

The ViewPoint application is capable of being expanded, so your installation can add extra functions as needed to suit your own special needs for distributed systems management. These functions include:

- Adding custom filters to filter events displayed on the Alternate Events screen.
- Adding descriptive text to the event-detail database for display on the Event Detail screen.
- Adding TACL macros and routines to automate tasks at your ViewPoint terminals.
- Altering the prefix text displayed for individual events on the Primary, Alternate, and Last Events screens.
- Determining the procedure for recovering events that takes effect after there is a CPU failure or after ViewPoint is shut down.
- Adding Extras screens for functions specific to your application that can be requested with function key SF15.
- Adding items to the status items displayed on the Network Status Summary screen.
- Configuring ViewPoint to run with the DSM/Problem Manager (DSM/PM) product.
- Installing ViewPoint to run as part of the Integrated Operations Console (IOC) product set. IOC consists of the following products: NetCommand, NetStatus, DSM/PM, and ViewPoint.

Each of these functions is described in Section 6, Customizing ViewPoint. Because the functions involve understanding of event management and programming for distributed systems management, you should also be familiar with the *SPI Programming Manual* and with the *EMS Manual*.

6 Customizing ViewPoint

Introduction

The ViewPoint application is an extensible Pathway application that you can use as a foundation for your software. This section describes the ways you can extend or customize the ViewPoint software for your system and network management needs. It describes how to:

- Add custom event-message filters and process operator-supplied filter parameters.
- Add advisory text to be displayed for individual event messages on the Event Detail screen.
- Alter the prefix text that is displayed for individual events on the Primary, Alternate, and Last Events screens.
- Determine whether or not the events in the cache will be retrieved when ViewPoint is restarted following a CPU failure or an intentional shutdown.
- Add TACL macros and routines to be started from the TACL screen (if the ViewPoint application provides access to TACL).
- Add Extras screens to provide additional functions including the capability to control deletion of events.
- Manage the addition of items to the status display.
- Obtain information about installing and using ViewPoint with the DSM/Problem Manager (DSM/PM) product.
- Obtain information about installing and using ViewPoint as part of the Integrated Operations Console (IOC) product set. IOC consists of the following products: NetCommand, NetStatus, DSM/PM, and ViewPoint.

Before using the information in this section, you should be thoroughly familiar with the features and architecture of the ViewPoint application as described in previous sections of this manual. You should also refer to Section 7, Installation, Configuration, and Startup for information on configuring your customized application.

If you are writing custom event-message filters, you should be familiar with the event-management filter language and compiler, as described in the *EMS Manual*. If you are adding Extras screens, you should be familiar with Pathway SCREEN COBOL programming. If you are adding items to the status displays, you should be familiar with writing Pathway servers.

Note. This section describes most of the ViewPoint software interfaces that use the DDL source code that is used in building the ViewPoint product. This DDL source code is included in a file that is distributed with the ViewPoint product. The file named GDDL contains the DDL. Companion files—GTAL, GCOB, and GLNK—provide TAL, COBOL85, and SCREEN COBOL Linkage Section versions of the declarations. Once you have installed the ViewPoint application, these files reside on the ViewPoint Installation Subvolume.

Writing Custom Event-Message Filters

The primary EMS **filter** and any optional alternate filters installed in the ViewPoint application determine which event messages are selected for display. ViewPoint provides two default filters on the installation subvolume: one for the Primary Events screen and the other for the Alternate Events screen.

The default filters tell the ViewPoint application what event messages to display at the node. Furthermore, they indicate to the ViewPoint software which event messages are flagged by their subsystems as representing **action events** (events requiring operator intervention) or potentially **critical events**, so that these can be highlighted as such in the displays.

You can tailor the ViewPoint event-message displays by writing custom event-message filters for use with any of the event presentation screens. Thus, you can arrange for selection of only a subset of the available event messages. The *EMS Manual* gives full instructions for writing filters using the EMS filter language.

The default event-message filter for the Primary Events and Last Events screens is the same for all users of the ViewPoint application. You can write a custom filter that differs from the default filter provided by ViewPoint and install this filter when ViewPoint is installed, replacing the default primary-events filter. Or, you can change the filter after ViewPoint is installed by stopping ViewPoint, changing the ViewPoint configuration, then restarting ViewPoint. (Refer to Section 7, Installation, Configuration, and Startup for a description of installing a new primary filter.)

If you have many messages going into the main ViewPoint console, you can separate the application messages from the system messages by configuring a separate event collector, besides the default, ($0), for all application messages. You will need two consoles, one for the filtered system messages, and one for the application messages because a ViewPoint terminal can not look at both at the same time.

Unlike the primary-events filter, the alternate-events filter that controls events sent to Alternate Events screens can be customized from within the ViewPoint application. There are several ways to do this.

You can write any number of alternate-events filters in addition to the default alternate-events filter provided by ViewPoint. This allows operators to choose a particular filter for a particular purpose. Once a custom filter exists, the operator can request that filter by specifying its name on the Alternate Event Configuration screen. (Refer to Section 3, Definition of ViewPoint Screens for a description of the Alternate Event Configuration screen.)

You do not have to write a custom alternate-events filter to control the events selected or discarded by the default filter. The default alternate-events filter contains code that allows an operator to configure the filter dynamically by entering parameters on the Alternate Event Configuration screen. (Refer to the subsection “Filter Parameters for Dynamic Configuration” later in this section for more information.) In addition, when you write your own filters, you can use the default filter as a model to include or extend this code (see “The Default Alternate-Events Filter” later in this section for the filter source code).

As an aid to writing custom filters, the next four subsections provide:

- Basic rules for the default filters provided with the ViewPoint application
- Source code for the default primary-events filter
- Source code for the default alternate-events filter
- A description of how the default alternate-events filter uses dynamic filter-configuration parameters

Basic Default Filtering Rules

The default primary-events and alternate-events filters follow certain basic rules. In both filters, ViewPoint passes all events that are not specifically suppressed and highlights action and critical events. The basic default filtering rules are:

1. First, the default filter checks for events that are not to be displayed on the ViewPoint screen—events with a SUPPRESS^DISPLAY token value of TRUE. ViewPoint always suppresses the display of messages for these events with one exception.

   The exception is for action-completion events—events that indicate that the action requested by an action event message has been taken. If the suppressed event is an action-completion event, an event with an ACTION^NEEDED token value of FALSE, the filter passes the event with a pass code of 3. When ViewPoint receives an action-completion event (an event with pass code 3), it does not display an action-completion message, but dims the corresponding action-needed message and decreases the event count by 1.

2. Next, the default filter checks for action events—events with an ACTION^NEEDED token. If the ACTION^NEEDED token is present, the filter sends ViewPoint a pass code of 1. When ViewPoint receives an action event (an event with pass code 1), it displays the event message highlighted as an action event.

3. Then, the default filter checks for critical events—events with an EMPHASIS token value of TRUE. If the EMPHASIS token has a value of TRUE, the filter sends ViewPoint a pass code of 2. When ViewPoint receives a critical event (an event with pass code 2), it displays the event message highlighted as a critical event.

4. If the event falls through the previous checks, it is an ordinary event (neither action nor critical). In this case, the filter does not send ViewPoint a pass code. When ViewPoint receives an event without a pass code, it displays an unhighlighted event message for that event.

Table 6-1 lists the corresponding ViewPoint action for each PASS value that ViewPoint receives.

Table 6-1. Pass Values and Corresponding ViewPoint Action

Pass Value	ViewPoint Action
0	Displays an unhighlighted event message for that event.

Table 6-1. Pass Values and Corresponding ViewPoint Action

Pass Value	ViewPoint Action
1	Displays the event message highlighted as an action event.
2	Displays the event message highlighted as a critical event.
3	Dims the corresponding action-needed message and decreases the event count by 1.

The default primary-events filter follows the basic rules listed above in determining what to display on the Primary Events and Last Events screens. The default alternate-events filter contains additional filtering rules that use operator input from the Alternate Events Configuration screen to help determine what event messages ViewPoint displays and how they are configured. The meaning of the filter parameters that can provide this dynamic configuration is described later in this section under the heading "Filter Parameters for Dynamic Configuration."

The source code for each of the default filters is provided in the next two figures. You can use this code as a basis for writing your own filters or for modifying the default filters.

The Default Primary-Events Filter

Figure 6-1 shows the source code for the default primary-events filter and the default last-events filter (FLTRDFLT and FLTRLAST, respectively). These filters are identical and are installed on the ViewPoint program subvolume. The source code file for the filter (SVPTFLTR) resides on the installation subvolume. The source code for this filter is shown in Figure 6-1.

This default filter source code filters out those events that have a SUPPRESS^DISPLAY token value of TRUE and that are not action-completion events (events with an ACTION^NEEDED token value of FALSE). In addition, it identifies action and critical events for use on the Primary Events screen.

Figure 6-1. Default Primary-Events Filter Source Code

```
FILTER ViewPoint^Default^Filter;
BEGIN

[#DEF zems^val^ss TEXT
  |BODY|[zspi^val^TANDEM].[zspi^ssn^zems].0]

BEGIN SSID(zems^val^ss)

== fails on suppress^display events which are not
== action-completion.

  IF zems^tkn^suppress^display = [zspi^val^true] THEN
    BEGIN
    IF TOKENPRESENT (zems^tkn^action^needed) AND
       zems^tkn^action^needed = [zspi^val^false] THEN
      PASS 3
    ELSE
      FAIL;
    END;

  == passes action-attention and action-completion
  IF TOKENPRESENT (zems^tkn^action^needed) THEN PASS 1;
  == testing for <> false for tape subsystem's sake
  IF zems^tkn^emphasis <> [zspi^val^false] THEN PASS 2;

  END;
PASS;
END;
```

The Default Alternate-Events Filter

The default alternate-events filter (FLTRALT) is installed on the ViewPoint program subvolume. The source code file for this filter (SFLTRALT), shown in Figure 6-2, resides on the installation subvolume. The default alternate-events filter passes all events unless parameters are provided. Passed events are identified (if appropriate) as action or critical for use on the Alternate Events screen.

△ **Caution.** If you want to write a custom filter, be sure to include the default source code shown in Figure 6-2 (below). This default code correctly processes any default parameters you might specify on page 2 of the Alternate Events Configuration Screen. If you do not include this default code and you attempt to process a default parameter, an error message might occur. See error message 0315 in Appendix B, Error Messages. (Page 4 of Figure 6-2 shows where you insert your custom code within the default source code.)

Figure 6-2. Default Alternate-Events Filter Source Code (page 1 of 4)

```
------------------------------------------------------------------------------
----------
-- ViewPoint default alternate event filter.                  --
-- This filter processes parameters which may be specified    --
-- from the second page of the Alternate Events               --
-- Configuration screen.  ZEMS and ZVPT TACL definition       --
-- must be loaded to compile this filter.                     --
------------------------------------------------------------------------------
----------

[#def zems^val^ssid text |body|[zspi^val^TANDEM].[zspi^ssn^zems].0]
[#def zvpt^val^ssid text |body|[zspi^val^TANDEM].[zspi^ssn^zvpt].0]

[#DEF init^template^key STRUCT
   BEGIN
   STRUCT fields;
     BEGIN
     SSID   template^ssid  value [zvpt^val^ssid];
     INT2   template^code  value [zvpt^tkn^inittemplate];
     INT2   template^value value [zvpt^val^inittemplate^null];
     END;
   BYTE  numbers(0:19) REDEFINES fields;

   END;
 ]

[#def InsertCommas macro |body|
 [#delta/commands ed^InsertCommas/%*%]
]

[#def ed^InsertCommas delta |body|
J<:S $;-DI,$>
]
------------------------------------------------------------
-- We need to #def our own replica of an ssid structure.
-- This is necessary because the one in ZSPITACL does not
-- have the fields broken out; instead that structure just
-- consists of a single field of data type ssid.  We need
-- to refer to the fields separately in this filter.
------------------------------------------------------------
 [#def zspi_ddl_ssid STRUCT
    CHAR Z^OWNER(0:7);
    INT  Z^NUMBER;
    UINT Z^VERSION;
    END;
 ]
```

Figure 6-2. Default Alternate-Events Filter Source Code (page 2 of 4)

```
FILTER Viewpt^alt^default^filter (
  ssid(zvpt^val^ssid, zvpt^fltr^select       )  OPTIONAL,
  ssid(zvpt^val^ssid, zvpt^fltr^discard      )  OPTIONAL,
  ssid(zvpt^val^ssid, zvpt^fltr^systemname   )  OPTIONAL,
  ssid(zvpt^val^ssid, zvpt^fltr^ssid         )  OPTIONAL,
  ssid(zvpt^val^ssid, zvpt^fltr^eventnumber  )  OPTIONAL,
  ssid(zvpt^val^ssid, zvpt^fltr^cpu          )  OPTIONAL,
  ssid(zvpt^val^ssid, zvpt^fltr^pin          )  OPTIONAL,

  ssid(zvpt^val^ssid, zvpt^fltr^processname  )  OPTIONAL,
  ssid(zvpt^val^ssid, zvpt^fltr^eventtext    )  OPTIONAL,
  ssid(zvpt^val^ssid, zvpt^fltr^custom^number)  OPTIONAL,
  ssid(zvpt^val^ssid, zvpt^fltr^custom^file  )  OPTIONAL,
  ssid(zvpt^val^ssid, zvpt^fltr^custom^string)  OPTIONAL );
begin ssid(zvpt^val^ssid)
if zvpt^fltr^select = [zspi^val^true] or
    zvpt^fltr^discard = [zspi^val^true]  then
   begin
   -- Test for matching system name (if provided).
   if ( not tokenpresent(zvpt^fltr^systemname) or
        zvpt^fltr^systemname =
           decompose(ssid(zems^val^ssid,zems^tkn^xsenderid^pd),system name)
      ) and

   -- Test for matching subsystem id (if provided).
      ( not tokenpresent(zvpt^fltr^ssid) or
        (  zvpt^fltr^ssid.zspi_ddl_ssid:z^owner =
             ssid(zems^val^ssid,zspi^tkn^ssid).
             zspi_ddl_ssid:z^owner
           and zvpt^fltr^ssid.zspi_ddl_ssid:z^number =
             ssid(zems^val^ssid,zspi^tkn^ssid).
             zspi_ddl_ssid:z^number
        )
      ) and

   -- Test for matching event number (if provided).
      ( not tokenpresent(zvpt^fltr^eventnumber) or
        zvpt^fltr^eventnumber =
      ssid(zems^val^ssid,zems^tkn^eventnumber) )

      and

   -- Test for matching cpu (if provided).
      ( not tokenpresent(zvpt^fltr^cpu) or
        zvpt^fltr^cpu =
        ssid(zems^val^ssid,
             zems^tkn^xsenderid).zems^ddl^xsenderid:zcpu

      ) and

   -- Test for matching pin (if provided).
      ( not tokenpresent(zvpt^fltr^pin) or
        zvpt^fltr^pin =
        ssid(zems^val^ssid,
             zems^tkn^xsenderid).zems^ddl^xsenderid:zpin
```

Figure 6-2. Default Alternate-Events Filter Source Code (page 3 of 4)

```
      ) and

   -- Test for matching process name (if provided).
      ( not tokenpresent(zvpt^fltr^processname) or
        decompose(zvpt^fltr^processname,destination name,name part) =
        decompose(ssid(zems^val^ssid,zems^tkn^xsenderid^pd),
                  destination name,name part)

      ) and

   -- Test for matching event text (if provided).
      ( not tokenpresent(zvpt^fltr^eventtext) or
        emstextmatch((
         [insertcommas [init^template^key:numbers(0:19)]]),
                     zvpt^fltr^eventtext)
      )

      --
      --  and ...
      --  user may add custom conditions here.
      --
   then
      -- conditions specified were met.
      begin
      if  zvpt^fltr^discard = [zspi^val^true] then fail;
      end

   else
      -- conditions specified were not met
      begin
      if  zvpt^fltr^select = [zspi^val^true] then fail;
      end;

   end; -- select or discard conditions

------------------------------------------------------------
-- Events which have passed the conditions above will be
-- processed using Viewpoint's default filtering rules:
-- 1) events with a suppress^display token value equal to
--    true will fail unless the event is an action
--    completion event -- then it will pass with a pass
--    code value of 3.
-- 2) events with an action^needed token will be passed
--    with a pass code of 1.
-- 3) events with an emphasis token value which is not
--    false will be passed with a pass code 2.
-- 4) All other events pass with no pass code.
------------------------------------------------------------

begin ssid(zems^val^ssid)   -- zems ssid section.
-- fail on suppress^display events which are not
-- action-completion.

if zems^tkn^suppress^display = [zspi^val^true] then
   begin
   if tokenpresent (zems^tkn^action^needed) and
      zems^tkn^action^needed = [zspi^val^false] then pass 3
```

Figure 6-2. Default Alternate-Events Filter Source Code (page 4 of 4)

```
   else
      fail;
   end;

-- events which have not been discarded by above conditions
-- are returned to Viewpoint.  Action and critical events
-- are identified by the following conditions:

-- action events are identified by the action^needed token.
if tokenpresent (zems^tkn^action^needed) then pass 1;
-- critical events are identified by the emphasis token.
if zems^tkn^emphasis <> [zspi^val^false] then pass 2;

end;   -- zems ssid section.
pass;
end;
```

You can write a custom filter to add IF conditions to select a subset of the available event messages. Example 6-1 shows a filter example that is designed to pass only those event messages that originate from three Compaq subsystems: Pathway, SNAX, and Expand.

Example 6-1. Filter to Pass Selected Event Messages

```
  -- Load ZSPI and ZEMS definitions
[SINK [#LOAD $system.ZSPIDEF.ZSPITACL]]
[SINK [#LOAD $system.ZSPIDEF.ZEMSTACL]]

   -- Initialize subsystem IDs, if this is not already done
[#DEF zpwy^val^ssid STRUCT
   BEGIN SSID x VALUE TANDEM.[zspi^ssn^zpwy].0; END;]
[#DEF zsx1^val^ssid STRUCT
   BEGIN SSID x VALUE TANDEM.[zspi^ssn^zsx1].0; END;]
[#DEF zexp^val^ssid STRUCT
   BEGIN SSID x VALUE TANDEM.[zspi^ssn^zexp].0; END;]

   -- Pass event messages from Pathway, SNAX, and EXPAND
FILTER select3;
BEGIN
  IF  zspi^tkn^ssid = SSID( zpwy^val^ssid )
  OR  zspi^tkn^ssid = SSID( zsx1^val^ssid )
  OR  zspi^tkn^ssid = SSID( zexp^val^ssid )  THEN
     BEGIN
     IF TOKENPRESENT (SSID(zems^val^ssid,
        zems^tkn^action^needed))  THEN PASS 1;
     IF  zems^tkn^emphasis = [zspi^val^true]  THEN PASS 2;
     PASS;
     END;
  FAIL;
END;
```

Filter Parameters for Dynamic Configuration

You can specify filter parameter values that dynamically change the default alternate-events filter (FLTRALT). Enter these values on page 2 of the Alternate Event Configuration screen. (Refer to the description of the Event Configuration screen in Section 3, Definition of ViewPoint Screens.) When you use this screen for alternate events, it displays alterable fields that show the current parameter values and allows you to enter new values. These parameters allow filtering criteria to be changed dynamically without requiring filter compilation or reconfiguration of ViewPoint.

A SELECT-DISCARD operand in the filter determines what action is to be taken on qualifying events. For example, if the operator chose SELECTED and then specified Subsystem ID as TANDEM.Pathway and Event Number as 3001, only Pathway events numbered 3001 would pass the filter. Conversely, if you chose DISCARDED and then specified the same parameter values, all events except Pathway events numbered 3001 would pass the filter.

The parameters allow the operator to select or discard events based on system name, subsystem ID, event number, CPU, process ID, or the event text. All the parameter values except event text are compared to the corresponding values of the process reporting the event. Event text is compared with event text produced by EMSTEXT. The entire event text (less header text) is used in the comparison. The parameter value provided for event text might contain an asterisk (*) or a question mark (?). A question mark (?) matches any one character; an asterisk (*) matches zero or more consecutive characters.

You can also alter the default alternate-events filter (FLTRALT)—or write a new filter—to change the logic or supply conditions that test for custom parameters. The operator specifies the custom parameters on the same screen as the other filter parameters. As many as four custom parameter values can be specified. Each value is one of the following types: file, string, or number (a 64-bit signed integer). These types correspond to the basic types supported by the EMS filter language. Specified values are passed to the filter in the following tokens: ZVPT^FLTR^CUSTOM^FILE, ZVPT^FLTR^CUSTOM^STRING, or ZVPT^FLTR^CUSTOM^NUMBER. Multiple instances of a single value type are represented as multiple-instance parameter tokens; indexes are used to refer to each instance.

If you create your own filter, it is not necessary for you to use the parameters specified on the Alternate Event Configuration screen. The default filter (FLTRALT) uses any noncustom parameters that are supplied, but you can change the configuration to use a filter that does not recognize these parameters. No warning will be issued at the time you process the screen.

Be aware that if you specify any of the custom parameter values and you have changed the configuration to use a filter that does not recognize these parameters, you probably will receive the error message "Error returned from EMS distributor:" (see error message 0315 in Appendix B, Error Messages).

Conversely, all parameter values are optional. If any of the parameter values are not specified (left blank), then the corresponding parameter token is not passed to the filter. If you want to use custom parameters, use the default filter shown in Figure 6-2 and include your custom source code at the location shown on page 4 of this figure.

Adding Advisory Text for Events

The Event Detail screen provides detailed information about a selected event message, including the probable cause of the event and the action recommended to solve the problem. This screen first displays identifying information about the event message, including the date and time the event was reported, the name of the process that reported the event, the subsystem ID of the reporting subsystem, the event number, and the text token if present.

The event information is followed by advisory text if entries exist for that event in the event-detail database. The advisory text is displayed under the headings “Cause,” “Effect,” and “Recovery.” This event-detail database consists of an EMS template file (EVENTTD) and a key-sequenced file (EVENTCX).

The EMS template file, EVENTTD, is installed in the ViewPoint program subvolume. It contains entries supplied by Compaq and can contain user-supplied entries. The file also allows for multiple-page text for events.

If you want to write customized text for the EVENTTD file, you must write a DDL dictionary for the template compiler and write EMS templates with cause, effect, and recovery.

△ **Caution.** Any customized text that you create for the EVENTTD file is overwritten by special releases. If there is an interim release, be sure to save a copy of your customized text.

Example 6-2 shows one way to write a DDL dictionary and Example 6-3 shows an example of an EMS template. For more information about writing customized text for the EVENTTD file, refer to the *DSM Template Services Manual.*

Example 6-2. DDL Dictionary for EVENTTD

```
? source zspiddl
? source zvptddl (zvpt-tkn-cause, zvpt-tkn-effect,
?                 zvpt-tkn-action)

definition zpwy-val-ssid tacl ssid.
 02 z-filler  type character 8 value is zspi-val-tandem
 02 z-owner redefines z-filler type zspi-ddl-char8.
 02 z-number type zspi-ddl-int. value is zspi-ssn-zpwy
 02 z-version type zspi-ddl-uint.
end
```

Example 6-3. EMS Template

```
VERSION: "T9153C20 - 15JUN88"
SSID:  ZPWY-VAL-SSID
SSNAME: "PATHWAY"
== Probable cause text for event 1034
MSG:  ZVPT-TKN-CAUSE,  10340001
"The following text is only for demonstration purposes:<*CR>"
"<*CR>"
"The dolphin is a marine mammal known for its intelligence and playfulness.
Dolphins are related to whales, but are smaller and have a beak-like snout.
Many people wonder about the difference between dolphins and porpoises;
porpoises are a type of dolphin.  Both belong to the order Cetacea.
Porpoises have blunt snouts and triangular dorsal fins.  An Orca (known as
Killer Whale) is a dolphin."
```

The other file, EVENTCX, can contain user-supplied entries. EVENTCX need not exist; however, if it does, it probably exists on the default program subvolume. It can, be placed anywhere if the CUSTOM-DETAIL server assign is used to refer to the file (see Appendix A, Server Assigns and Parameters for a description of the CUSTOM-DETAIL assign).

△ **Caution.** The entries in EVENTCX override the matching Tandem entries in EVENTTD.

You use the EVENTCX file to add “Cause” and “Recovery” advisory text to be displayed about specified event messages. To do this, you must build the EVENTCX file and then add records containing the appropriate text to the file. Once this is done, the text you supplied appears in the display whenever the operator selects the Event Detail screen for an event message of that type.

By customizing your advisory text, you can allow the operator to look up the probable causes and recovery actions for Compaq subsystem events as they apply to your particular system or network. For instance, an advisory display for an event message requesting the operator to mount another tape could include specific information on tape mounting procedures at your installation. Likewise, you can allow the operator to see advisory text for events reported by a non-Compaq application—for example, startups and shutdowns of automatic teller machines (ATMs).

EVENTCX Record Structure

The DDL shown in Figure 6-3 describes the records for the EVENTCX file. Refer to the *SPI Programming Manual* for a more thorough description of the fields within the subsystem ID, but note that the version number is not included in the SUBSYSTEM-ID definition in EVENTCX. For Compaq subsystems, event numbers are compatible from release to release. Subsystems you write should also follow this convention.

Note. The Z-NUMBER definition might generate a different range of values depending on the language that you use. TAL generates a 16-bit value, and SCOBOL generates a 4-digit value no longer than 9999. Refer to the *Pathway SCREEN COBOL Reference Manual* for more details about this value range.

Figure 6-3. DDL for Event-Detail Database

```
definition  ZVPT-EVENT-SSID.
    03 Z-OWNER       type character 8.
    03 Z-NUMBER      type binary 16.
 end

record ZVPT-EVENT-DETAIL.
file is "EVENTCX" key-sequenced.

* field identifying the event
 02 Z-EVENT.
    03 Z-EVENT-SSID         type ZVPT-EVENT-SSID.
    03 Z-EVENT-NUMBER       type binary 16.

* fields providing 6 lines of probable cause text
 02 Z-PROBABLE-CAUSE.
    03 Z-LINE              pic x(77) occurs 6 times.

* fields providing 6 lines of recommended action text
 02 Z-RECOMMENDED-ACTION.
    03 Z-LINE              pic x(77) occurs 6 times.

 key is Z-EVENT.
end
```

An Enable Application to Generate Event-Detail Records

You can use this DDL description to produce a standard Enable application that generates event-detail records. To do this, use the following procedure:

1. Compile the DDL shown in the previous figure and create a FUP command file:

   ```
   DDL /IN DDLSRC/ DICT, FUP FUPSRC !
   ```

2. Execute the FUP file-creation statements in FUPSRC:

   ```
   FUP /IN FUPSRC/
   ```

3. Establish a command file for the Pathway configuration:

   ```
   EDIT ENABEX
   *ADD
   1    PURGE ENABLOG, ENABCTL
   2    CREATE ENABLOG
   3    ASSIGN PATHCTL, ENABCTL
   4    PATHMON /NAME $PM, NOWAIT, OUT ENABLOG/
   5    PATHCOM /IN ENABPATH/ $PM
   6    PATHCOM $PM; RUN DETAIL-PROG
   7    PATHCOM $PM; SHUTDOWN, WAIT
   8    //
   *EXIT
   ```

4. Call Enable, set attributes, and generate the application:

```
ENABLE

%SET RECORD ZVPT-EVENT-DETAIL
%SET PATHCOMFILE ENABPATH !
%GENERATE DETAIL-PROG
%EXIT
```

5. Execute the command file and run the application:

```
OBEY ENABEX
```

6. Enter your custom advisory text from the keyboard using the Enable application screen interface.

Altering Prefix Text for Events

You can alter the prefix text that appears with event messages on the Primary, Alternate, and Last Event screens. To do this, specify the EVENT-PREFIX parameter in the PATHDEFS configuration file. The parameter has three possible settings: 1, 2, and 3. The default setting is 1. If you use setting 1, the prefix will contain the time of the event and the name of the system that generated it; the prefix will be in the format:

```
hh:mm \sysname.
```

hh

the hour of day that the event was generated.

mm

the minute in the hour that the event was generated.

sysname

the name of the system that generated the event.

If you use setting 2, the prefix is displayed with the date and time of the event and the system name; it will be in the format:

```
mon-dd hh:mm \sysname.
```

mon-dd

the month and day of the year in which the event was generated.

hh

the hour of day that the event was generated.

mm

the minute in the hour that the event was generated.

sysname

the name of the system that generated the event.

If you use setting 3, he prefix is displayed with the time of the event, the system name, and an identifying character (!, * or blank) in the beginning of the prefix to indicate the type of event (critical, action, or other); it will be in the format:

```
^hh:mm \sysname...
```

^

^ is the ! character (to indicate critical events), the * character (to indicate action events), or a blank character (to indicate all other events).

hh

the hour of day that the event was generated.

mm

the minute in the hour that the event was generated.

sysname

the name of the system that generated the event.

Any changes you make to the EVENT-PREFIX parameter are changed after you restart ViewPoint. For more information about setting the EVENT-PREFIX parameter, see Appendix A, “Server Assigns and Parameters.”

You can also create a custom event prefix by modifying the existing event template used by ViewPoint. The prefix specified in the template is used by all ViewPoint terminals running on the system. You can find the event template source code used to create the template in subvolume $*vol*.ZTEMPL where $*vol* is the volume on which the ZTEMPL ISV was installed. The template is normally found in the system template file on $SYSTEM. For help with creating, compiling, and installing templates, see the *DSM Template Services Manual*.

Recovering Events After a Shutdown

If there is a CPU failure affecting the ViewPoint event collection server or if ViewPoint is intentionally shut down, the ViewPoint event display cache is lost. You can determine whether or not these events are recovered when ViewPoint is restarted by setting a parameter called RECOVER-CACHE. This parameter determines the kind of recovery procedure that occurs automatically after the failure or shutdown.

The two possible settings are True and False (the default setting). False means that no events previously listed in the cache will be recovered. When ViewPoint is started

again, the primary and alternate event displays configured to monitor events begin displaying events as they occur; any events previously stored in the events cache are lost. You get the best performance by setting the RECOVER-CACHE parameter to False.

True means that all events previously contained in the event cache are retrieved. When ViewPoint is started again, the primary and alternate event displays begin displaying events starting with the the oldest event contained in the cache before the shutdown.

Adding TACL Macros and Routines

If your ViewPoint application provides a TACL interface, the operator can use the TACL screen to run programs, including TACL macros and routines. Using TACL and the Define Process feature, you can write macros and routines for the operator to start from the TACL screen.

Define Process allows you to define many background requesters that continue running so you can pass commands to them and control the output. This allows faster response time because, for many commands, a majority of the time spent executing the command is spent starting up the process.

Using a single interactive process to perform a command on a large number of objects—for instance, to start a large number of terminals or data-communications lines—can be time-consuming, since a single process operates on only one object at a time. Using the Define Process feature, you can write a TACL macro or routine that defines many instances of a subsystem management process such as SCF; sends commands to each process for different objects; and waits for any one or all of the commands to finish, taking full advantage of the multiprocessing environment of the Compaq system. Your macro or routine can send commands from a file or a variable and queue the commands up for various processes. It can display the resulting output, delete it, or save it in a file or variable.

In addition, you can write a macro, routine, or obey file to set up background processes—that is, start tasks to be done automatically while the operator is using other ViewPoint features, such as the various status and event-message screens. Background tasks performed by such a routine could include starting and stopping measurements using the Measure performance measurement tool, generating reports periodically with the Enform query language, and running regular batch jobs at specified times. For information on the Define Process features that permit setting up background processes, refer to Section 4, “Process Definition Commands.”

Example 6-4 illustrates three simple TACL macros. The first TACL macro starts background PATHCOM, CMI, and DNSCOM processes. The second lists status information from each. The third stops the background processes.

Example 6-4. Sample TACL Macros for Background Processes

```
?SECTION dpstart MACRO
==**  dpstart
==**  Start PATHCOM, CMI, and DNSCOM.
DP PATHCOM /PNAME pcom/ $ZVPT
DP CMI /PNAME cm/
DP DNSCOM /PNAME dnsc/

?SECTION dpstatus MACRO
==**  dpdns
==**  List status of PATHMON, CMP, and DNS.
pcom STATUS PATHMON
cm STATUS CMP
dnsc STATUS

?SECTION dpstop MACRO
==**  dpstop
==**  Stop PATHCOM, CMI, and DNSCOM.
UNDP pcom cm dnsc
```

Note. Before executing these macros, you must indicate that the Process Definition commands are in the :UTILS:DP directory. To do this, you can enter the following command at the TACL prompt:

```
USE :UTILS:DP
```

Or, you can include this command in your logon TACLCSTM file.

Example 6-5 shows a TACL routine to examine all event messages that the operator has saved in the ViewPoint clipboard file, query the Distributed Name Service (DNS) for information on the device names in the event message, and display the resulting information. The routine DNSINFO calls another routine named UTILS:DNSINFO, which returns DNS information on the device names in one line from the clipboard file.

Example 6-5. Sample TACL Routine to Examine ViewPoint Clipboard

```
?SECTION utils:dnsinfo ROUTINE
==** [utils:dnsinfo <event text>]
==* Look up info on all devices named in <event text>.
#FRAME
#PUSH :^device :^dnsout
[#LOOP |WHILE| [#MORE] |DO|
  [#IF ([#ARGUMENT /VALUE :^device/
              FILENAME WORD/SPACE/] = 1) |THEN|
    [#IF ([#MATCH \*.$* [:^device]] or
              [#MATCH $* [:^device]]) |THEN|
      dnsc /OUTV :^dnsout/ INFO [:^device]
      [#IF NOT [#EMPTYV :^dnsout] |THEN|
        #OUTPUT Device [:^device]:
        #OUTPUTV :^dnsout
        #SET :^dnsout ]
    ]
  ]
]
#UNFRAME

?SECTION dnsinfo ROUTINE
==** DNSINFO
==* Look at all event messages in the clipboard
==* and get DNS info for each.
#FRAME
#PUSH :^clipboard :^line :^clipfilename
#IF [#ARGUMENT END]
#SET :^clipfilename [#DEFAULTS /SAVED/].zzvpclip
[#IF NOT [#FILEINFO /EXISTENCE/ [:^clipfilename]] |THEN|
  == no clipboard file
  #RESET FRAMES
  #RETURN
]
[#IF NOT [_PCHECK /EXISTENCE/ dnsc] |THEN|
  DP DNSCOM /PNAME dnsc, QUIET/
]
#SET /IN [:^clipfilename]/ :^clipboard
PURGE [:^clipfilename]
[#LOOP |WHILE| NOT [#EMPTYV :^clipboard] |DO|
  #SET :^LINE [#EXTRACT :^clipboard]
  [#IF NOT [#EMPTYV/BLANK/ :^line] |THEN|
    #OUTPUT Event: [:^line]
    utils:dnsinfo [:^line] ]
]
#UNFRAME
```

Adding Extras Screens

As provided by Compaq, the ViewPoint application includes predefined screens for TACL, system or network status, and event display. However, you can extend the

ViewPoint application to include custom screens specific to your application or to display screens of other applications.

The **Extras** feature allows you to add a custom screen by writing a SCREEN COBOL program to drive that screen, along with any Pathway servers needed to perform the functions the operator can request from the new screen. Once you have configured your custom SCREEN COBOL program and servers, the extras screen and its functions become part of your ViewPoint application.

Furthermore, you can include additional custom screens, accessible by function keys from the first custom screen, by adding the necessary SCREEN COBOL programs and Pathway servers. For instance, you might want your initial Extras screen to be a menu from which the operator can select any one of several custom screens, each of which allows the operator to send a different command to a subsystem. The screen for each command could include fields for entering the command parameters.

Linkage Section of ZVPT-EXTRAS

On standard ViewPoint screens, where the Extras function key is provided, if you press this key, the result is a call to a SCREEN COBOL program unit named ZVPT-EXTRAS. If the ZVPT-EXTRAS program unit exists, it takes control of the terminal. If the program unit does not exist, ViewPoint displays an error message. If you do not want the EXTRAS program to be executed, you must delete it from the program object library.

Note. The ViewPoint program object files provided by Compaq do not include a ZVPT-EXTRAS program unit. You should build a separate object library to contain your ZVPT-EXTRAS program unit.

Your ZVPT-EXTRAS program unit must have the following PROCEDURE DIVISION definition:

```
PROCEDURE DIVISION
  USING ZVPT-IPC-HDR,
        ZVPT-LINK-DATA-L,
        ZVPT-USER-REQUEST.
```

When the ViewPoint application calls the ZVPT-EXTRAS program, it passes it a SCREEN COBOL Linkage Section containing context information. This Linkage Section is shown in Figure 6-4.

Figure 6-4. Linkage Section of ZVPT-EXTRAS

```
 01 ZVPT-IPC-HDR.
    02 Z-REQUEST-CODE                  pic s9(4)    comp.
    02 Z-PW-REPLY-CODE redefines Z-REQUEST-CODE
                                       pic s9(4)    comp.
    02 filler                          pic x.
    02 Z-VERSION-CODE.
      03 LETTER                        pic a.
      03 REV-NUMBER                    pic 99.
    02 Z-IPC-RETN-CODE                 pic s9(4)    comp.
    02 Z-IPC-RETN-CODE-DETAIL          pic s9(4)    comp.
    02 Z-LOG-THIS-IPC                  pic a.
    02 filler                          pic x.

  01 ZVPT-LINK-DATA-L.
    02 Z-TMF-STARTED-FLAG              pic 9.
      88 Z-TMF-STARTED                 value is 1.
    02 Z-COLOR-FLAG                    pic 9.
      88 Z-COLOR                       value is 1.
    02 Z-SYSTEM-NAME                   pic x(8).
    02 Z-PATHMON-NAME                  pic x(8).
    02 Z-TACL-PORT-NAME                pic x(24).
    02 Z-PRINT-PORT-NAME               pic x(34).
    02 Z-CLIPBOARD-FILENAME            pic x(24).
    02 Z-CURRENT-VOL-SUBVOL            pic x(16).
    02 Z-STATUS-CONFIG-FILENAME        pic x(24).
    02 Z-EVENT-CONFIG-FILENAME         pic x(24).
    02 Z-USER-ID.
      03 Z-USER-ID-GROUP               pic s9(4)    comp.
      03 Z-USER-ID-USER                pic s9(4)    comp.
    02 Z-HELP-SERVER                   pic x(16).
    02 Z-PREFERRED-LANGUAGE            pic x(32).

 01 ZVPT-USER-REQUEST.
    02 Z-LAST-SCREEN                 pic x(16).
    02 Z-FUNCTION-KEY                pic x(16).
    02 Z-OPTION-LINE                 pic x(78).
```

The information in ZVPT-IPC-HDR and ZVPT-LINK-DATA-L forms the context for the thread of execution on a particular terminal. Your ZVPT-EXTRAS program can use this information, but the program should not alter the information. One exception is where a terminal is running without a TACL connection. In that case, you might want to change fields such as Z-CLIPBOARD-FILENAME, Z-CURRENT-VOL- SUBVOL, and Z-USER-ID to correspond to the user's environment, if the information is available to you.

The information in ZVPT-USER-REQUEST includes the options specified on entry to ZVPT-EXTRAS (the contents of the options line on a block-mode screen or, if Extras was called from TACL, the contents of the input following the TACL prompt). You can use these fields to invoke a different base ViewPoint screen on exit from ZVPT-EXTRAS and to pass a different option line to that screen.

Descriptions of the pertinent fields follow. Fields shown in Figure 6-8 that are not mentioned below exist for compatibility reasons; your ZVPT-EXTRAS program unit should not alter those fields or rely upon them for information.

`Z-VERSION-CODE of ZVPT-IPC-HDR`

provides the identifying letter (LETTER) and number (REV-NUMBER) indicating the version of the ViewPoint software that is running.

`Z-COLOR-FLAG of ZVPT-LINK-DATA-L`

indicates whether or not the ViewPoint application was invoked with the color option. A value of 1 in this field denotes the color option. Your ZVPT-EXTRAS program can use this information to select color or monochrome displays.

For a color display, the default ViewPoint colors are white characters on a blue background, with special attributes to produce green action event messages and red critical event messages. If you prefer a different combination of foreground and background colors, use #push and #set for one or both of the following TACL variables:

```
:utils_globals:ViewPoint:_foreground color
:utils_globals:ViewPoint:_background color
```

where *color* can be black, blue, green, cyan, red, magenta, brown, or white.

`Z-SYSTEM-NAME of ZVPT-LINK-DATA-L`

is two to eight characters identifying the system on which the ViewPoint application is running. The first character is always a backslash (\). This field is padded on the right with blanks.

`Z-PATHMON-NAME of ZVPT-LINK-DATA-L`

is two to six characters identifying the process name of the PATHMON process under which the ViewPoint application is running. The first character is always a dollar sign ($). This field is padded on the right with blanks.

`Z-CLIPBOARD-FILENAME of ZVPT-LINK-DATA-L`

is the file name of the clipboard file for the current operator. This file name is in internal form, ready to be passed to the file-system OPEN procedure. Servers invoked from the ZVPT-EXTRAS program can read or update this file. The file is created by the first “copy to clipboard” function. It might not exist if no “copy to clipboard” function has been invoked. Note that by default, servers run using the process access ID of the PATHMON process associated with the ViewPoint application. Clipboard files are created with the default file security of the current operator, unless you are running ViewPoint without a TACL connection (in which case the ViewPoint PATHMON user ID security is used). Server file operations on the clipboard file might yield security errors if the access ID and file security are incompatible. For more information about the clipboard feature, refer to the discussion of function keys in Section 3, “Definition of ViewPoint Screens,” and also to Section 2, “Using ViewPoint.”

```
Z-CURRENT-VOL-SUBVOL of ZVPT-LINK-DATA-L
```

is the current default volume and subvolume for the TACL process associated with the terminal. The volume and subvolume names are in internal form (16 characters). This field is used to resolve file names. It corresponds to the user's default subvolume as of the last time the TACL screen was entered. If no TACL connection is present, this field contains the name of the ViewPoint data subvolume. If this is not the appropriate subvolume, the field can be changed by the ZVPT-EXTRAS program.

```
Z-STATUS-CONFIG-FILENAME of ZVPT-LINK-DATA-L
```

is the file name of the current status configuration file. This file name is in internal form. It is the same as would be displayed by the Profile screen.

```
Z-EVENT-CONFIG-FILENAME of ZVPT-LINK-DATA-L
```

is the file name of the current event configuration file. This file name is in internal form. It is the same as would be displayed by the Profile screen.

```
Z-USER-ID of ZVPT-LINK-DATA-L
```

is the user ID of the operator who is currently logged on to the ViewPoint application. This user ID is in internal form, consisting of two binary numbers—the group ID and the user ID. This value changes if the user ID is changed from the TACL screen.

```
Z-LAST-SCREEN of ZVPT-USER-REQUEST
```

is one to sixteen characters identifying the screen from which the Extras screen was invoked. The value is padded on the right with blanks. The following list shows the possible values of this field and the corresponding invoking screens. (Note that in these values, case is significant.)

```
"Status"          Network Status Summary screen
"Pri-Event"       Primary Events screen
"Alt-Event"       Alternate Events screen
"Last-Event"      Last Events screen
"TACL"            TACL conversational screen
"Profile"         Profile screen
```

```
Z-FUNCTION-KEY of ZVPT-USER-REQUEST
```

is one to sixteen characters identifying the function (screen) to be invoked. The value is padded on the right with blanks. On entry to the ZVPT-EXTRAS program unit, the value is "Return." If the value is not modified, then on exit from the ZVPT-EXTRAS program unit, the old screen (from which the Extras screen was invoked) is reestablished. If the value is changed, a new function is invoked on return from ZVPT-EXTRAS. The following list shows the possible values of this field and identifies the screen that is invoked when each value is moved into Z-FUNCTION-KEY. (Note that in these values, case is significant.)

```
"Status"          Network Status Summary screen
"Pri-Event"       Primary Events screen
```

```
"Alt-Event"      Alternate Events screen
"Last-Event"     Last Events screen
"TACL"           TACL conversational screen
"Profile"        Profile screen
"Exit"           Exit the ViewPoint application for
                 this terminal
```

`Z-OPTION-LINE of ZVPT-USER-REQUEST`

is optional text to be passed to the invoked screen. On entry, this contains the text that was on the option line when the Extras screen was invoked. If the Extras screen was invoked from the conversational TACL screen, the value is the text entered on the option line before the Extras function key was pressed. The text is padded on the right with blanks.

On exit, if the Z-FUNCTION-KEY field is set, your program should set Z-OPTION-LINE to whatever text should be passed to the new screen as an option. If the Z-FUNCTION-KEY field is not modified, Z-OPTION-LINE is ignored.

If the Extras function key is pressed from the Primary Events, Alternate Events, or Last Events screen, any marked events are also passed to the ZVPT-EXTRAS program unit. Events passed are those marked by a nonblank character in column 1 or by cursor position. These events are passed as event messages in SPI format in a data file described under “Passing Marked Event Messages” below.

Your ZVPT-EXTRAS program unit can call other program units you write. It should not call program units delivered with the ViewPoint product. Program units you write can send commands to server classes your application creates. They should not send commands to server classes delivered with the ViewPoint product.

Passing Marked Event Messages

Event messages that are marked on an Events screen when the Extras function key is pressed are passed to the ZVPT-EXTRAS program unit. Event messages are marked by a nonblank character in column 1 or by cursor position on the screen. Only messages from the current screen are passed. Event messages are passed as messages in SPI format in an ENSCRIBE key-sequenced data file (EVNTMRKD) in the ViewPoint data subvolume.

EVNTMRKD is shared among all terminal threads and contains a record for each event message passed. The key for each record includes the logical terminal name and an event count (1, 2, 3, and so on). The logical terminal name portion of the key can be used to position (generic position) on event messages passed for a given terminal. The event count distinguishes successive event messages marked on a screen. The header structure on each record includes the filter return code and a deleted flag.

On entry to a ZVPT-EXTRAS program unit, messages for events marked on the ViewPoint Events screen are present in EVNTMRKD. After returning to ViewPoint from a ZVPT-EXTRAS program unit, all event records corresponding to the logical terminal are automatically deleted from EVNTMRKD.

The DDL in Figure 6-5 shows the EVNTMRKD record structure. This DDL is distributed with ViewPoint in the file GDDL (with language versions in the files GTAL for TAL, GCOB for COBOL85, and GLNK for SCREEN COBOL).

Figure 6-5. DDL for Marked Event Database (EVNTMRKD)

```
definition ZVPT-MARKED-KEY.
* Key of record within Event-Marked file.
   02 Z-LOGICAL-TERM         pic x(16).
   02 Z-EVENT-COUNT          type binary 16.
end.

definition ZVPT-EVENT-MARKED.
   02 ZVPT-MARKED-KEY        type *.
   02 Z-FILTER-RETURN-VALUE  type binary 16.
     88 Z-IS-NORMAL-EVENT    value 0.
     88 Z-IS-ACTION-EVENT    value 1.
     88 Z-IS-CRITICAL-EVENT  value 2.
   02 Z-DELETED-FLAG         type binary 16.
   02 Z-EVENT-BUFFER-LEN     type binary 16.
   02 Z-EVENT-BUFFER         pic x occurs 2 to 3900 times
                             depending on
                             Z-EVENT-BUFFER-LEN.
 end.

 record ZVPT-MARKED-REC.
  file is "EVNTMRKD" key-sequenced.
  definition is ZVPT-EVENT-MARKED.
  key        is ZVPT-MARKED-KEY.
  end.
```

Descriptions of the pertinent fields follow.

`Z-LOGICAL-TERM of ZVPT-MARKED-KEY`

is a string of one to fifteen alphanumeric characters or hyphens identifying the logical terminal name of the Pathway thread. This value corresponds to the value of the SCREEN COBOL special register, LOGICAL-TERMINAL-NAME.

`Z-EVENT-COUNT of ZVPT-MARKED-KEY`

is an ordinal number (1, 2, 3, and so on) identifying successive events marked for a single logical terminal.

`Z-FILTER-RETURN-VALUE of ZVPT-EVENT-MARKED`

is the integer value passed with the event when the event filter was applied. If no value was passed, this value is 0 (zero). If a ViewPoint default filter was used, this value is 1 for action events, 2 for critical events, 3 for action-completion events, and 0 for all other events.

`Z-DELETED-FLAG of ZVPT-EVENT-MARKED`

is a Boolean value indicating whether an event was deleted. TRUE (-1) indicates that the event was an action or critical event, and that it was deleted either automatically or by operator action. FALSE (0) indicates that the event was not deleted.

`Z-EVENT-BUFFER-LEN of ZVPT-EVENT-MARKED`

is the event message length in bytes.

`Z-EVENT-BUFFER of ZVPT-EVENT-MARKED`

is an SPI buffer containing the event message. This is a variable-length field, with a maximum of 3900 bytes. For how to retrieve tokens or text from the event message in this buffer, refer to the *EMS Manual*.

EXTRAS-DELETE Interface

Optionally, ViewPoint can call a user-provided program unit whenever an operator attempts to manually acknowledge an event message from the Primary Events screen. The called program unit can authenticate or audit the acknowledge operation using the user ID and the event buffers. An array of flags returned in the Linkage Section of the program unit is used to permit or deny acknowledge operations on the marked event messages. The program unit must be named ZVPT-EXTRAS-DELETE and should have the following PROCEDURE DIVISION definition:

```
PROCEDURE DIVISION
  USING ZVPT-IPC-HDR,
        ZVPT-LINK-DATA-L,
        ZVPT-USER-REQUEST,
        ZVPT-DELETE-ACCEPTED.
```

The Linkage Section declarations for ZVPT-EXTRAS-DELETE are identical to those for ZVPT-EXTRAS shown earlier, with one additional declaration: ZVPT-DELETE-ACCEPTED. Note, however, that the link back to ViewPoint differs from that of ZVPT-EXTRAS. ZVPT-EXTRAS can call any other ViewPoint screen; ZVPT-EXTRAS-DELETE can exit and return only to the Primary Events screen.

ZVPT-DELETE-ACCEPTED contains an array of Boolean flags; each flag corresponds to an event message marked for deletion. On entry to the program unit, the flags all have a value of TRUE, indicating permission to acknowledge. If a flag is set to FALSE (0), the corresponding event message that is marked for deletion is not deleted and ViewPoint displays an advisory message on line 20 of the Events screen. The ZVPT-DELETE-ACCEPTED declaration is shown in Figure 6-6 and is distributed with ViewPoint in the file GDDL (with language versions in GTAL for TAL, GCOB for COBOL85, and GLNK for SCREEN COBOL).

Figure 6-6. DDL for ZVPT-DELETE-ACCEPTED

```
definition ZVPT-DELETE-ACCEPTED.
  02 Z-DELETE-ACCEPTED-ALL-FLAGS  pic x(50).
  02 Z-DELETE-ACCEPTED-FLAGS      pic 9 occurs 50 times
              redefines Z-DELETE-ACCEPTED-ALL-FLAGS.
     88 Z-DELETE-ACCEPTED         value 1.
  02 Z-REASON                     pic x(66).
end.
```

Event messages selected for acknowledgment are passed as messages in SPI format in a data file. (This file was described earlier under the subsection “Passing Marked Event Messages.”) The order of events in the data file corresponds to the order of the selected event messages on the Events screen. This is also the order of the flags in ZVPT-DELETE-ACCEPTED. Hence, event 1 in the data file corresponds to Z-DELETE-ACCEPTED-FLAGS (1) in ZVPT-DELETE-ACCEPTED and corresponds to the first event message marked for deletion on the screen.

A single SCREEN COBOL statement denies the acknowledge operation for all marked event messages:

```
MOVE ZEROES TO z-delete-accepted-all-flags.
```

The program unit can supply an explanatory message for ViewPoint to display when the acknowledge operation is denied. The message should be moved into the Z-REASON field of ZVPT-DELETE-ACCEPTED. ViewPoint displays the contents of this field on line 20 of the Events screen if an acknowledge operation is denied and Z-REASON is not blank.

The option to call ZVPT-EXTRAS-DELETE is indicated by a Boolean parameter (CALL-EXTRAS-DELETE) that you must add to the server configuration for ZVPT-EVNT-COLL in the PATHBASE file used to configure ViewPoint. TRUE indicates that the program unit ZVPT-EXTRAS-DELETE should be called; FALSE (the default value) indicates that this program unit should not be called. (Refer to Section 7, Installation, Configuration, and Startup for a description of changing the PATHBASE configuration file.)

Preparing Your ZVPT-EXTRAS Program for Installation

You must compile all the program units you write into an object library that is referred to by the TCLPROG routine of the ViewPoint program. This library is initially named POBJ on the ViewPoint program subvolume. You can invoke the compilation as follows:

```
SCOBOLX /IN source-file, OUT $S.#HOLD, NOWAIT/ POBJ
```

During installation of the ViewPoint application, ViewPoint program units are copied to the POBJ library on the ViewPoint program subvolume.

Server classes that you refer to in your custom program units must either be added to the Pathway application associated with the ViewPoint application or else exist within a separate Pathway application. You use a SEND verb that specifies PATHMON and SYSTEM to send a command to a server class in a separate Pathway application. Refer

to the *Pathway SCREEN COBOL Reference Manual* for instructions on using SEND to an externally defined server class.

For information on how to add server classes to the Pathway application and on how to configure a ZVPT-EXTRAS-DELETE program unit into the ViewPoint application, refer to Section 7, Installation, Configuration, and Startup.

Custom Status Servers

The Network Status Summary screen gives an overview of the status of the system or network. As configured by Compaq, this screen displays summary information on several types of objects: CPUs, disks, terminals, and data-communications lines. The displays give two different kinds of information: counts (such as the number of units up or down) and busy values (indicating that a unit is *n* `percent` busy over a measured period of time). For count values, the count is used to calculate a percentage of the configured count and this percentage is displayed graphically. For busy values, a unit-busy percentage is retrieved (using Measure) and displayed.

A **status item** is a specified measurement of a particular type of object (or in some cases, of a particular object of a given type, such as a particular CPU), either as a count or a busy value. The standard status server provided with the ViewPoint application can display the following status items:

- CPU busy
- Average CPU busy (within a system)
- Disk busy
- Line busy
- Count of Pathway terminals running (per Pathway system)
- Count of TMF transactions per second (obtained from TMFSERVE)
- Count of SNAX LUs started
- Count of SNAX PUs started
- Count of SNAX lines started

For the busy status items, the operator or system manager specifies the names of the devices to be monitored using the Status Item Configuration screen.

A name, called a **display type**, is associated with each status item. Operators can read the display types for status items on the Status Item Configuration screen.

Low and high thresholds can be configured for each item being measured. These thresholds define a condition that causes the item to be highlighted on the screen for emphasis. Each status item consists of a server name, a display type for that server, a user name for the item, an optional 100 percent value, threshold values, and an option to reverse the significance of the data received from the server. (For more information on these capabilities of the Network Status Summary screen, refer to Section 3, Definition of ViewPoint Screens.)

You can customize the Network Status Summary screen to display other status items in addition to the standard items. These can include information from subsystems you write, such as information about the operation of automatic teller machines (ATMs). They can also include information from Compaq subsystems, such as the batch subsystem, that are not monitored by the basic ViewPoint application.

Status Servers

The ViewPoint application includes a server that collects data for the standard status items. To allow the Network Status Summary screen to display information on additional items, you can write one or more custom status collection servers, or **status servers**, to augment or replace the work of the default Compaq status server.

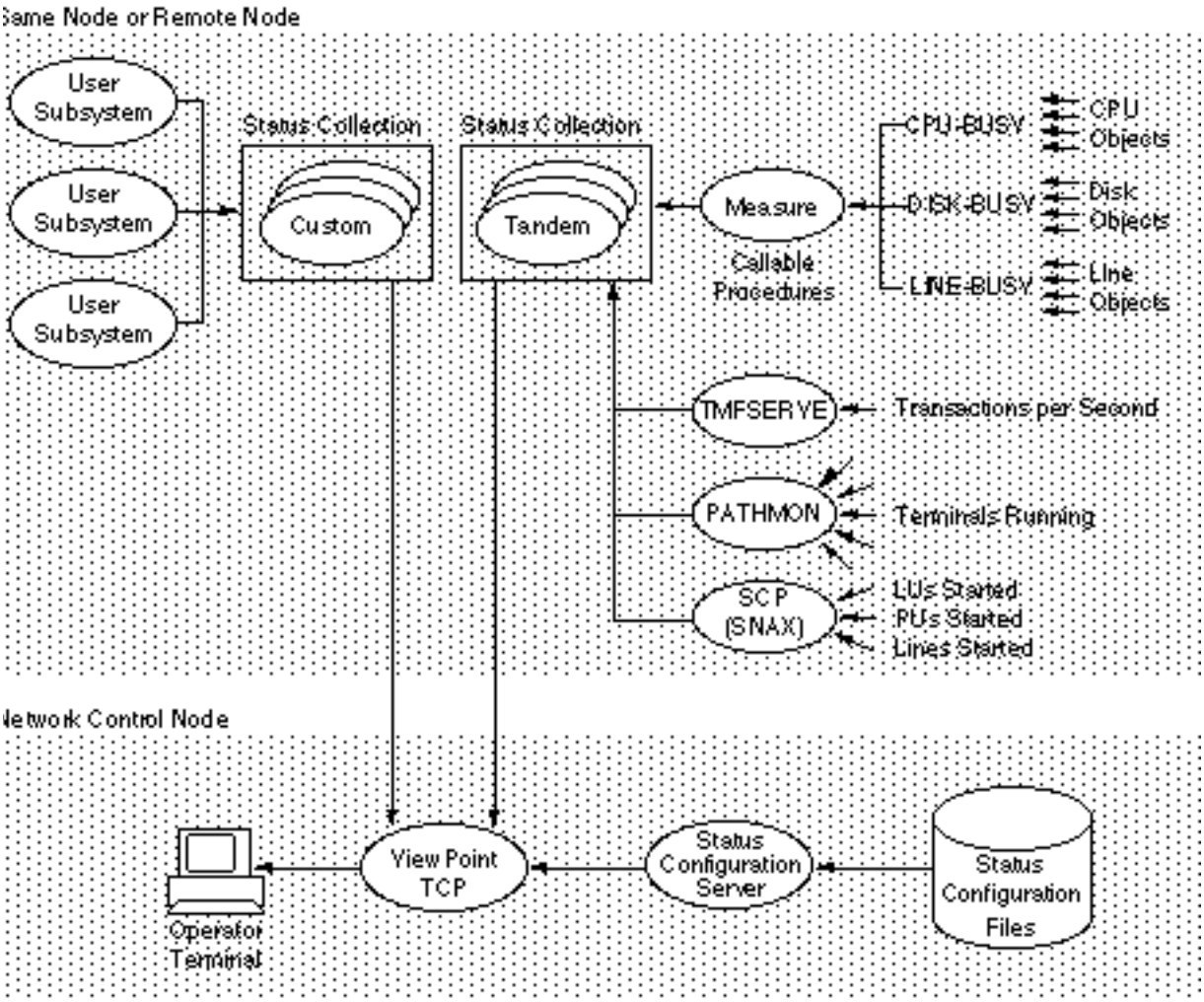
When you write a status server, you should writer the server to be a standard Pathway context-free server that is defined to the ViewPoint application or to another Pathway system as a dynamic server class. You do not need to write the server as a process pair. You can write the server in any language acceptable to Pathway, but your server can use SPI only if you write it in TAL, COBOL85, or TACL.

A status server is identified by a server-class name, which is specified from the Status Item Configuration screen. The standard status server delivered with the ViewPoint product is installed with the server-class name ZVPT-STAT-COLL. Status servers you write must have a different server-class name.

Server classes can be added to the Pathway application associated with the ViewPoint application or they can exist in separate Pathway applications. The procedure for adding a server class to the Pathway application associated with the ViewPoint application is described in Section 7, Installation, Configuration, and Startup.

The various processes involved in status collection are illustrated in Figure 6-7.

Figure 6-7. Example of Status Reporting Configuration



Commands for and Responses From the Status Server

The ViewPoint application maintains one or more status configuration files, one for each operator who is currently using the ViewPoint application. Each of these files lists all the status items, along with associated option values, such as reverse significance, that the corresponding operator has currently specified for display using the Status Item Configuration screen. Each item listed in a status configuration file includes the name of the status server used to retrieve information on that item. The operator, through the Status Item Configuration screen, directs the ViewPoint software to update the status configuration files. To access the facilities of a new status server, the operator must know the name of the new server and the item names it recognizes. Status servers need not be concerned with the structure of the status configuration files.

The task of the status server is to perform certain functions on behalf of the ViewPoint requester software. As it updates its status configuration or display and requires a

certain function to be performed, the ViewPoint software sends a message to the status server requesting that function. The status server performs the specified function, then returns a response message to the requester.

The ViewPoint requesters send six types of commands to the status server. The commands are:

- SCAN-ITEM and NEXT-ITEM
  Ask the server for information about status items that the server supports. (Each status server must maintain a list of all the status items it supports, with default option values for each.)
- CHECK-ITEM
  Asks the server to check the configuration of a status item that is about to be added.
- SAMPLE-ITEM
  Asks the server to retrieve current data for a status item.
- SAMPLE-LIST
  Asks the server to return information on several status items with one request. Retrieving data on status items is the task most frequently performed by the status server.
- GET-VERSION
  Asks the server to identify the message interface version that the server implements.

The SCAN-ITEM, NEXT-ITEM and CHECK-ITEM commands are created for the Status Item Configuration screen when the operator presses the corresponding function keys. The SAMPLE-ITEM and SAMPLE-LIST commands are created for the Network Status Summary screen after the items have been configured. (Refer to Section 3, Definition of ViewPoint Screens for information about these screens.) The GET-VERSION command is issued prior to any SAMPLE-LIST command to ensure that the version is one that supports SAMPLE-LIST.

Each command message consists of a header structure, a command code, and command data. Each response message consists of a header structure and response data. Figure 6-8 shows the DDL for the command and response message headers, which are common to all messages.

Figure 6-8. Status-Server Message Header

```
definition ZVPT-IPC-HDR.
  02 Z-REQUEST-CODE                type binary 16.
  02 Z-PW-REPLY-CODE redefines Z-REQUEST-CODE
                                   type binary 16.
     88 Z-RQST-OK                  value is 0.
     88 Z-RQST-ERR                 value is 3.
  02 filler                        pic x.
  02 Z-VERSION-CODE.
     03 Z-LETTER                   pic a.
     03 Z-REV-NUMBER               pic 99.
  02 Z-IPC-RETN-CODE               type binary 16
                                   value 0.
  02 Z-IPC-RETN-CODE-DETAIL        type binary 16
                                   value 0.
  02 Z-LOG-THIS-IPC                pic a.
  02 filler                        pic x.
end.

definition Z-REQUEST-CODE          type binary 16.
```

Descriptions of the pertinent fields follow. Fields not mentioned exist for compatibility reasons; your server should not use or alter those fields.

`Z-PW-REPLY-CODE of ZVPT-IPC-HDR`

is the Pathway response code. It must be set to 0 (indicating no errors) or 3 (indicating that an error occurred). The error code (if any) is set in Z-IPC-RETN-CODE.

`Z-VERSION-CODE of ZVPT-IPC-HDR`

on entry, is the software release version of the command message structure; on exit, it is the release version of the response message structure. The server should set the version code to the version of GDDL that it is using, even if an error is being returned.

`Z-IPC-RETN-CODE of ZVPT-IPC-HDR`

is set to 0 (zero) or to an error code to be returned. Figure 6-9 lists the allowed error-code values.

`Z-REQUEST-CODE`

is present in all commands, immediately following the message header. It contains a code indicating the type of command made. Figure 6-9 lists the possible values of this command code.

Figure 6-9 gives the DDL for the command-code values that ViewPoint sends to the server and the error-code values that the server can return.

Figure 6-9. Status-Server Command Codes and Error Codes

```
* REQUEST-CODE VALUES

CONSTANT Z-GET-VERSION-CODE           VALUE IS 1.
CONSTANT Z-SCAN-ITEM-CODE             VALUE IS 2.
CONSTANT Z-NEXT-ITEM-CODE             VALUE IS 3.
CONSTANT Z-CHECK-ITEM-CODE            VALUE IS 4.
CONSTANT Z-SAMPLE-ITEM-CODE           VALUE IS 5.
CONSTANT Z-SAMPLE-LIST-CODE           VALUE IS 6.

* ERROR-CODE VALUES

CONSTANT Z-ALL-OK-CODE                 VALUE IS 0.
CONSTANT ZERR-STAT-COLLECT-REQUEST     VALUE IS 0206.
CONSTANT ZERR-STAT-NO-SUCH-TYPE        VALUE IS 0207.
CONSTANT ZERR-STAT-NO-SUCH-OBJECT      VALUE IS 0208.
CONSTANT ZERR-STAT-END-OF-TYPES        VALUE IS 0209.
CONSTANT ZERR-STAT-CNFG-INCOMPATIBLE   VALUE IS 0210.
CONSTANT ZERR-STAT-OBJECT-UNAVAILABLE  VALUE IS 0211.
```

Status-Server Command Codes

This section gives a general description of the command that corresponds to each command code. The structures of the command and response messages are shown in a figure in the subsection “Message Structures for Status Commands and Responses.”

`Z-GET-VERSION-CODE`

This command retrieves the version of the message interface that the server implements. The server returns the software release version of the GDDL file it is using (for instance, “C20” for the C20 release) in Z-VERSION-CODE of ZVPT-IPC-HDR.

`Z-SCAN-ITEM-CODE`

This command provides a display-type field; its value can be blank or nonblank. If it is blank, the response should be information about the first status item implemented by the server. If the value is not blank, the server should search all status items it implements for a matching display type. If a match is found, the server should return information about that status item in the response. If no match is found, the server should return the error code ZERR-STAT-NO-SUCH-TYPE. When the response contains status-item information, it should also contain an index into the list of status items that the server implements. If the first status item is returned, this index value should be 1.

`Z-NEXT-ITEM-CODE`

This command provides the index of a previously scanned status item. The server should increment this index and return the status item to which the new index value corresponds. If the new index value exceeds the number of status items

implemented by the server, the server should return the error code ZERR-STAT-END-OF-TYPES.

`Z-CHECK-ITEM-CODE`

This command provides a status item that has been specified from the Status Item Configuration screen. The server should check the status item for consistency. Possible error codes returned are ZERR-STAT-NO-SUCH-TYPE, ZERR-STAT-NO-SUCH-OBJECT, and ZERR-STAT-CNFG-INCOMPATIBLE.

If no errors are detected, the server should return the status item with changes (if any) that the server wants to enforce. Two fields that the server should always set are Z-SCALE and Z-VALUE-TYPE. These fields cannot be changed from the Status Item Configuration screen.

`Z-SAMPLE-ITEM-CODE`

This command provides a previously configured status-item display type and object and expects a response containing the current value, the maximum value, and a timestamp. Possible errors are ZERR-STAT-OBJECT-UNAVAILABLE, ZERR-STAT-NO-SUCH-TYPE, and ZERR-STAT-NO-SUCH-OBJECT.

`Z-SAMPLE-LIST-CODE`

This command provides a list containing configured status items (type and object) and expects in reply a list containing the current value, maximum value, and timestamp for each item in the request list. This command allows several items to be sampled with one request. Each sample in the list can generate an error; the reply includes an error field for each entry in the list. Possible errors are ZERR-STAT-OBJECT-UNAVAILABLE, ZERR-STAT-NO-SUCH-TYPE, and ZERR-STAT-NO-SUCH-OBJECT.

Status-Server Error Codes

The error codes given previously in Figure 6-9 are described below. As shown in Figure 6-9, the nonzero errors correspond to error numbers 206 through 211 that are displayed by the ViewPoint application. When the server detects one of the errors described, it assigns the proper code to Z-IPC-RETN-CODE in the response-message header. Z-PW-REPLY-CODE is assigned a value of 3. Only the response header is included in the response message.

`Z-ALL-OK-CODE`

The status server detected no errors.

`ZERR-STAT-COLLECT-REQUEST`

The command to the status server was unrecognizable (because the command code was not provided or was invalid).

ZERR-STAT-NO-SUCH-TYPE

The display type provided in Z-ITEM-NAME of ZVPT-STATUS-ITEM (see Figure 6-14 and the discussion following it) did not match any that are implemented by this status server.

ZERR-STAT-NO-SUCH-OBJECT

The object name provided was not a valid object for the display type provided.

ZERR-STAT-END-OF-TYPES

The SCAN-NEXT command was made, but scanning has progressed past the last status item that is supported by this status server.

ZERR-STAT-CNFG-INCOMPATIBLE

A CHECK-ITEM command was made, but the status-item configuration values given were incompatible with the display type and object name.

ZERR-STAT-OBJECT-UNAVAILABLE

A SAMPLE-ITEM or SAMPLE-LIST command was made, but the object named in the command was unavailable. The most likely reason is that the process (such as PATHMON) that manages the object was not running.

Status Items

The server must retain a list of the status items it supports, giving for each item the default values of various display options. (The ViewPoint displays use these default values unless the operator changes them from the Status Item Configuration screen.) The server searches this list when it receives the SCAN-ITEM command, and steps through it on successive NEXT-ITEM commands. In the list, each status item is represented by a structure called ZVPT-STATUS-ITEM (shown in Figure 6-10). This structure is included in several of the command and response messages, as shown in a figure later in the subsection “Message Structures for Status Commands and Responses.”

Figure 6-10. Status-Server Structures

```
definition ZVPT-STATUS-VALUE  type binary 64.

definition ZVPT-TIMESTAMP     type binary 64.

definition ZVPT-BOOLEAN       pic x.

definition ZVPT-ITEM-NAME.
   02 Z-ITEM-TYPE          pic x(30).
   02 Z-OBJECT-NAME        pic x(24).
end.

definition ZVPT-STATUS-ITEM.
   02 Z-ITEM-DESCRIPTION   pic x(36).
   02 Z-ITEM-NAME          type ZVPT-ITEM-NAME.
   02 Z-VALUE-FOR-100      type ZVPT-STATUS-VALUE.
   02 Z-USE-MAXIMUM        type ZVPT-BOOLEAN.
   02 Z-REVERSE-SIGNIFY    type ZVPT-BOOLEAN.
   02 Z-SCALE              type binary 16.
   02 Z-LOW-THRESHOLD      type binary 16.
   02 Z-HIGH-THRESHOLD     type binary 16.
   02 Z-LOW-ENABLED        type ZVPT-BOOLEAN.
   02 Z-HIGH-ENABLED       type ZVPT-BOOLEAN.
   02 Z-VALUE-TYPE         pic xx.
      88 Z-COUNTER-VALUE   value is "CV".
      88 Z-BUSY-TIMER      value is "BT".
end.
```

The server should return the ZVPT-STATUS-ITEM structure in response to SCAN-ITEM commands. The values in ZVPT-STATUS-ITEM fields in the response message become the default values for the fields. Some of the fields can be changed from the Status Item Configuration screen. The CHECK-ITEM command allows the server to check these changed values for consistency before they are added to the configuration file. The fields of ZVPT-STATUS-ITEM are described in the following paragraphs.

`Z-ITEM-DESCRIPTION`

This is the description provided on the screen when an item is being displayed. It should be padded on the right with blanks. The server should provide a default description. There is no need to check this value.

`Z-ITEM-NAME`

This field (consisting of the subfields Z-ITEM-TYPE and Z-OBJECT-NAME) identifies the item. Z-ITEM-TYPE is the display type for the status item. Z-OBJECT-NAME is optional; it is unnecessary for certain display types, such as “average CPU busy.” When Z-OBJECT-NAME is not provided, it is blank. As a part of the CHECK-ITEM command, the server should check Z-OBJECT-NAME to see if it is a valid object for Z-ITEM-TYPE. Both values are padded on the right with blanks.

Z-VALUE-FOR-100

This is the value used to compute the percentage for counter-type values when Z-USE-MAXIMUM is FALSE. If the server allows Z-USE-MAXIMUM to be FALSE for an item, it should require that this field have a reasonable value when performing CHECK-ITEM for the item. If Z-USE-MAXIMUM is TRUE or if this is a busy-timer item, this field is ignored.

Z-USE-MAXIMUM

If this value is TRUE (“Y”) and this is a counter-value item, ViewPoint uses the Z-MAXIMUM-VALUE field in the SAMPLE-ITEM response to compute a percentage. If this value is FALSE (“N”), then ViewPoint uses Z-VALUE-FOR-100. If the item is a busy-timer item, this value is ignored.

Z-REVERSE-SIGNIFY

This value indicates “Y” for inverted or “N” for not inverted for the status display. You can use this value to convert an objects-up display into an objects-down display.

Z-SCALE

This value indicates the number of digits assumed to the right of a decimal point in status values. The value can be 0, 1, or 2. This value cannot be changed by configuration; the server should set it in the response to the CHECK-ITEM command. For most counters, Z-SCALE should be 0. If the item is a busy-timer item, this value is ignored.

Z-LOW-THRESHOLD and Z-HIGH-THRESHOLD

These values are percentage values (0-100). They are used as thresholds to trigger highlighting of the display item on the status display. These values are ignored if the respective ENABLED flags are FALSE.

Z-LOW-ENABLED and Z-HIGH-ENABLED

These are flags (“Y” or “N”) that indicate whether or not the respective thresholds are enforced. “Y” means they are enforced.

Z-VALUE-TYPE

This is a flag (“CV” or “BT”) that indicates the basic type of the status item: counter-value or busy-timer. This flag cannot be changed by configuration. The server should set it in the response to the CHECK-ITEM command.

Your status server should provide default values for the fields in the STATUS-ITEM structure when it returns the structure in response to the SCAN-ITEM and NEXT-ITEM commands. The operator can change these values from the Status Item Configuration screen; the server should check the changed values in response to the CHECK-ITEM command.

The SAMPLE-ITEM command must return status values; it returns the values in 64-bit binary entities. The two basic types of status values are busy timers and counters. The value sampled from a busy timer is the number of microseconds that the object was

busy. This value is a snapshot of the busy timer; successive snapshots are used to compute a percentage. Counters are scalar values that are used directly. They are a count of the number of objects up, or throughput counts.

Counter values can be fractions; a Z-SCALE field is included in the ZVPT-STATUS-ITEM structure. Its value (0, 1, or 2) indicates the number of decimal positions assumed in the count. Z-SCALE is a static property; it cannot be changed by configuration.

The Z-MAXIMUM-VALUE field is optionally set to contain the maximum value a counter can attain. This field is used to compute a percentage for the status display if the item was configured to use the maximum. Some counts, such as throughput counts, have no definite maximum values. A field called Z-USE-MAXIMUM in ZVPT-STATUS-ITEM is set during configuration to specify whether the Z-MAXIMUM-VALUE field is used. The server checks the value of this field in the CHECK-ITEM command, and chooses to reject the configured value of this field by returning ZERR-STAT-CNFG-INCOMPATIBLE. If the server accepts a TRUE value (“Y”) in this field, it should always set Z-MAXIMUM-VALUE in response to the SAMPLE-ITEM command. Counts of objects up are items for which Z-MAXIMUM-VALUE is useful. The Z-MAXIMUM-VALUE field can be ignored for busy-timer items.

A timestamp is supplied with each sample. This timestamp is the four-word, Julian-date-based, microsecond-resolution timestamp at the time of the sample (or as soon as possible after the sample), as returned by the JULIANTIMESTAMP system procedure call.

The SAMPLE-LIST command is the equivalent of one or more SAMPLE-ITEM commands. SAMPLE-LIST improves performance because it retrieves all the samples—up to a maximum of 16—from a given server with one command. A return code must be set for each of the 16 requested items. The return code is used in the same way that the Z-IPC-RETN-CODE is used in SAMPLE-ITEM; that is, 0 (zero) indicates no error. (Note that Z-IPC-RETN-CODE is still used in the SAMPLE-ITEM command.) If the value is nonzero, all samples received by the request are in error; the particular error is determined by the value of Z-IPC-RETN-CODE. If Z-IPC-RETN-CODE is 0 (zero), return codes for each status item determine the success or failure of the particular sample.

The SAMPLE-LIST command is sent to servers that implement version C20 of ViewPoint, as determined by the version value returned in the Z-VERSION-CODE field. A version value of C20 (or later) indicates that ViewPoint recognizes SAMPLE-LIST. The ViewPoint status server (ZVPT-STAT-COLL) recognizes SAMPLE-LIST.

Message Structures for Status Commands and Responses

The message structures shown in Figure 6-11 refer to definitions in previous figures: the message-header definitions (Figure 6-8) and the status-server structure definitions (Figure 6-10).

Note. Be sure to initialize to 0 (zero) all variables in the reply; this is particularly important for variables that are otherwise ignored such as Z-OBJECT-INDEX. Variables that are not initialized can cause unpredictable errors.

Figure 6-11. Status-Server Command and Response Messages (page 1 of 2)

```
Request/Reply Headers

definition ZVPT-REQUEST-HEADER.
   02 Z-IPC-HDR            type ZVPT-IPC-HDR.
   02 Z-REQUEST-CODE       type ZVPT-REQUEST-CODE.
end.

definition ZVPT-REPLY-HEADER.
   02 Z-IPC-HDR            type ZVPT-IPC-HDR.
end.

* Request/Reply Data (Tails) for the Status Server
*
* Definition GET-VERSION-REQUEST (no request data)
* Definition GET-VERSION-REPLY  (no reply data)

definition ZVPT-SCAN-ITEM-REQUEST.
   02 Z-ITEM-NAME          type ZVPT-ITEM-NAME.
end.

definition ZVPT-SAMPLE-LIST-REQUEST.
   02 Z-SAMPLE-COUNT       type binary 16.
   02 Z-SAMPLE-ITEM        type occurs 1 to 16 times
                           depending on Z-SAMPLE-COUNT.
      03 Z-ITEM-NAME       type ZVPT-ITEM-NAME.
end.

definition ZVPT-SCAN-ITEM-REPLY.
   02 Z-STATUS-ITEM        type ZVPT-STATUS-ITEM.
   02 Z-ITEM-INDEX         type binary 16.
   02 Z-OBJECT-INDEX       type binary 16.
end.

definition ZVPT-NEXT-ITEM-REQUEST.
   02 Z-ITEM-INDEX         type binary 16.
   02 Z-OBJECT-INDEX       type binary 16.
end.

definition ZVPT-NEXT-ITEM-REPLY.
   02 Z-STATUS-ITEM        type ZVPT-STATUS-ITEM.
   02 Z-ITEM-INDEX         type binary 16.
   02 Z-OBJECT-INDEX       type binary 16.
end.

definition ZVPT-CHECK-ITEM-REQUEST.
   02 Z-STATUS-ITEM        type ZVPT-STATUS-ITEM.
end.
```

Figure 6-11. Status-Server Command and Response Messages (page 2 of 2)

```
definition ZVPT-CHECK-ITEM-REPLY.
   02 Z-STATUS-ITEM        type ZVPT-STATUS-ITEM.
 end.

definition ZVPT-SAMPLE-ITEM-REQUEST.
   02 Z-ITEM-NAME          type ZVPT-ITEM-NAME.
end.

definition ZVPT-SAMPLE-ITEM-REPLY.
   02 Z-STATUS-VALUE       type ZVPT-STATUS-VALUE.
   02 Z-MAXIMUM-VALUE      type ZVPT-STATUS-VALUE.
   02 Z-TIMESTAMP          type ZVPT-TIMESTAMP.
end.

definition ZVPT-SAMPLE-LIST-REPLY.
   02 Z-SAMPLE-COUNT        type binary 16.
   02 Z-SAMPLE-ITEM         occurs 1 to 16 times
                            depending on Z-SAMPLE-COUNT.
      03 Z-RETN CODE        type binary 16.
      03 Z-RETN-CODE-DETAIL type binary 16.
      03 Z-STATUS-VALUE     type ZVPT-STATUS-VALUE.
      03 Z-MAXIMUM-VALUE    type ZVPT-STATUS-VALUE.
      03 Z-TIMESTAMP        type ZVPT-TIMESTAMP.
end.
```

The field Z-RETN-CODE indicates success (Z-ALL-OK-CODE) or an error (ZERR-STAT-*name*).

The command-message and response-message fields not previously described are now described below.

`Z-SCALE`

This value indicates the number of digits assumed to the right of a decimal point in status values. The value can be 0, 1 or 2. This value cannot be changed by configuration; the server should set it in the response to the CHECK-ITEM command. For most counters, Z-SCALE should be 0. If the item is a busy-timer item, this value is ignored.

`Z-ITEM-INDEX`

In SCAN-ITEM and NEXT-ITEM responses, this value is the index of the returned status item; index values begin at 1.

In NEXT-ITEM commands, this value is the index of the previously scanned item. The server adds 1 to this value to get the index of the next status item, then returns this next item.

`Z-OBJECT-INDEX`

This field exists for compatibility reasons; the server must initialize it to 0 (zero).

Integrating Your Status Server With the ViewPoint Application

To integrate your custom status server with the ViewPoint application, you first define the status server as a dynamic server class to the ViewPoint application or to another Pathway system. Then, you use the Status Item Configuration screen to identify each supported item, the name of the server class, and the values of the display parameters (if different from the default values). For details on configuration, refer to Section 7, Installation, Configuration, and Startup.

Refer to Appendix D, Sample Custom Status Server for the source code of a sample custom status server. This server collects status information for the ViewPoint Network Status Summary screen.

Using ViewPoint with DSM/PM

ViewPoint can be configured to run with the DSM/Problem Manager (DSM/PM) application. When configured to run with DSM/PM, ViewPoint allows a user with a valid DSM/PM user name to do the following:

- Log on to DSM/PM from any ViewPoint screen and perform any DSM/PM functions.
- Add and submit an event marked on a ViewPoint event screen as a problem to DSM/PM.
- Add, but do not submit, an event marked on a ViewPoint event screen as a problem to DSM/PM.
- Add and submit an event marked on a ViewPoint event screen as a problem information record to DSM/PM.

For information on configuring ViewPoint to run with DSM/PM, see the *DSM/Problem Manager Installation and Customization Guide*. For examples showing how to access DSM/PM from ViewPoint, see the subsection, “Using DSM/PM,” in Section 2, Using ViewPoint in this manual.

Using ViewPoint in the IOC

ViewPoint can be installed as part of the Integrated Operations Console (IOC) product set. In addition to ViewPoint, the IOC consists of the following products:

- NetCommand
  A command line interface for running utilities and application programs.
- NetStatus
  A full-screen interface for controlling and monitoring the status of objects in a Compaq network.
- DSM/PM
  A tool that automates problem management within a Compaq network.

With the IOC, you can easily access and move between each product, and perform any of the tasks you normally would with the stand-alone versions of each one.

7

Installation, Configuration, and Startup

This section describes the steps for installing, configuring, and starting ViewPoint. The procedures in this section must be completed before you can do the procedures shown in Section 2, Using ViewPoint.

ViewPoint Installation

The ViewPoint software is installed at your site with newly released Compaq software, using the Install program that is documented in the *System Generation Manual*. Following a cold load of the system image tape (SIT), which is produced by the Install program, some additional steps are necessary to complete the installation of ViewPoint. These additional steps are described in the section, "Installation Steps," later in this section. Before you begin installation, read the following subsections, "General Installation Information" and "Before You Begin."

General Installation Information

The overall installation procedure can be summarized as follows:

- The Install program moves a release of ViewPoint from tape to an Installation Subvolume (ISV).
- You then use a TACL routine (also called Install) to create ViewPoint installations from the Installation Subvolume (ISV).

The system Install program automatically installs all the necessary TACL routines for ViewPoint configuration, installation, and invocation and for installing the Define Process functions. TACL uses a shared directory called :UTILS to hold these TACL routines provided by Compaq.

The Install program places ViewPoint TACL routines under the three directories shown in Table 7-1.

Table 7-1. Three Directories for ViewPoint TACL Routines

Directory	Type of Routines
:UTILS:DP	Define Process
:UTILS:VIEWP	Installation and run-time
:UTILS:VPTLIB	Tools

You should customize your #USELIST to include these directories. See the subsection, "Before You Begin" later in this section for how to include these directories in your #USELIST.

The Install routine provided by ViewPoint automates the installation and configuration of ViewPoint by using information you supply to install and update necessary program, data, and configuration files. The files that are affected include:

- Program and configuration files for the ViewPoint Compaq *NonStop*™ TS/MP requesters and servers.
- Data files used by the NonStop™ TS/MP requesters and servers.
- NonStop™ TS/MP™ configuration files: PATHDEFS, PATHCONF, and PATHSTRT. The installation process produces these files, if the files do not already exist, and updates these files with values you enter.
  - PATHDEFS contains the commands to add the ViewPoint program and servers to a NonStop™ TS/MP configuration.
  - PATHCONF contains the commands to start your NonStop™ TS/MP system from a cold state and to add and start the ViewPoint TCP to the NonStop™ TS/MP configuration. PATHCONF invokes PATHDEFS.
  - PATHSTRT contains the commands to start ViewPoint servers.
- Compile-time files (DDL) for use by user-written applications that are added to the ViewPoint configuration. (These files are not moved from the ISV.)
- Definition file which directs files to their intended subvolumes and sets common configuration parameters.

Note. The Install program also allows you to change the command settings in your PATHDEFS and PATHCONF configuration files without also installing new program and data files. This is useful for updating your configuration files in between installations of new software. You do it by entering an option, CONFIGURE, when you first invoke Install.

The Install program follows some general guidelines when installing a software update:

- Files released by Compaq replace any old versions.
- Files containing user data are not replaced.
- If installation replaces an old copy of a file, the new file is given the owner and security of the old file.
- If installation creates a new file, the new file is owned by the user running the installation routine. The security attributes set for each file are “NUNU” with the exception of the TERMTACL file, which is set to “NNNU.”

Note. The maximum number of Alternate Event screens that can be configured on a single ViewPoint installation on a G-Series system is 70. The maximum number of Alternate Event screens that can be configured on a D-Series system installation is 50.

Before You Begin

Before installing ViewPoint, check to make sure you have read access to the ViewPoint ISV files. These ISV files take on the ownership and default security of the user who runs the system Install program that restores the SUT (site update tape) tape on your

system. To avoid a security error, make sure you check the security attributes of the ViewPoint ISV files.

To use the ViewPoint TACL routines :UTILS:DP, :UTILS:VIEWPT, and :UTILS:VPTLIB, without typing :utils every time, include these TACL routines in your use list. To do this for all users on your system, type the following commands in your TACLLOCL file ($SYSTEM.SYSTEM.TACLLOCAL). Or, you can add these commands to your TACLCSTM file to customize only your environment. The commands you include are:

```
> #push #informat      == save current value
> #set #informat TACL
> #set #uselist [#uselist] :utils:DP
 :utils:viewpt:utils:vptlib
> #pop #informat       == restore its value
```

Also, if you are installing ViewPoint onto a subvolume that contains an earlier version of ViewPoint, you might want to rename the copies of your PATHDEFS, PATHCONF, and PATHSTRT files because any existing copies of these files are erased during the installation. (ViewPoint prompts you before erasing the files to give you a chance to save them.) Later, you can refer to the settings in the old copies of PATHDEFS, PATHCONF, and PATHSTRT when you are updating the new versions of files.

Finally, all ViewPoint processes run at a high PIN as the default; the exceptions are: ZVPT-EVNT-COLL, ZVPT-EVNT-NTFY, and ZVPT-EVLE-COLL. These processes, as well as PATHMON and PATHTCP2, must be run at a low PIN if you are running ViewPoint under a mixed mode of C-series and D-series systems.

Installation Steps

To install ViewPoint using the ViewPoint Install TACL routine, follow these steps:

1. You have a choice of performing a full installation, which includes new program, data, and configuration files, or updating the ViewPoint configuration file.

   - To perform a full installation, which includes new program, data, and configuration files, type:

```
VIEWPT:INSTALL [filename]
```

filename

is the name of the define file. If you do not specify a define file, ViewPoint prompts you for a file name. If you have not created a define file, you can use the default file; press return when prompted for the file name.

Note. If you do not see the ViewPoint banner when you give this, your system might not be configured to point to the default search directory :UTILS. If you are not able to enter ViewPoint, retype this command using the full path name:

```
:UTILS:VIEWPT:INSTALL
```

- To update the ViewPoint configuration file, type:

```
VIEWPT:INSTALL /configure/ [filename]
```

filename

is the name of the define file. If you do not specify a define file, ViewPoint prompts you for a file name. If you have not created a define file, you can use the default file; just press return when prompted for the file name.

A startup screen (see Example 7-1) appears that describes the installation process and asks you to specify the name of the definition file you want to use.

Example 7-1. Installation Startup Screen

```
Viewpoint: install


Viewpoint - TP040020 - Installation procedure.
This procedure will build a pathway configuration and install
Viewpoint files from the installation subvolume. BREAK may be
pressed at any time to stop this procedure. Defaults are
displayed in square brackets and may be selected by pressing
RETURN.


Please specify the name of the definition file for this
installation. [$data.viewpt.define]
```

If you supply the name of an existing file, the Install program uses the values in that file to produce the option display. If the specified file does not exist, Install creates it.

If you want to use the default file name, press RETURN. Install displays the default file name, DEFINE, on the current subvolume.

1. When you are satisfied with the name of your DEFINE file, press RETURN to continue the installation process.

   The installation options now appear on your screen (see Figure 7-2). They are a list of the most commonly-used parameter values appearing in the PATHDEFS and PATHCONF configuration files. When you select a different option value, Install automatically changes the associated parameter value appearing in PATHDEFS or PATHCONF. At the same time it updates your define file, which contains a complete list of all the parameter settings.

   Note. If you specified an existing definition file when you first ran Install, the settings listed in the definition file are displayed on your screen. Otherwise, default settings are displayed.

Example 7-2. Installation Options

```
Viewpoint installation options:

1.  ISV                    $DSV.ZVIEWPT
2.  PROGRAM-SUBVOL         $DATA.VIEWPT
3.  DATA-SUBVOL            $DATA.VIEWPT
4.  PATHWAY-POBJ           $DATA.VIEWPT.POBJ
5.  EVENT-NOTIFY           $ZEVEN
6.  CPUS                   0:1
7.  SERVER-PRIORITY        DEFAULT
8.  TCP-PRIORITY           DEFAULT
9.  PRIMARY-COLLECTOR      $0
10. PRIMARY-FILTER         $DATA.VIEWPT.FLTRDFLT
11. LAST-EVENT-COLLECTOR   $0
12. LAST-EVENT-FILTER      $DATA
13. CLEANUP-HOUR           23
14. PRIMARY-CACHESIZE      320
15. LAST-EVENT-CASHESIZE   10
16. RECOVER-CACHE          FALSE
17. SupressUnanswered      FALSE
18. PATHWAY-OWNER          COMMSW.NSM
19. PATHWAY-SECURITY       "C"
20. PROGRAM-SECURITY       "N"

Enter a number and press RETURN for an explanation or to update
an option value. Press BREAK or CTRL-Y to exit without
installation. Enter "C" and press RETURN to continue ViewPoint
installation.
```

2. To change an option, type its number, enter a new value, and press RETURN. (You can interrupt the installation process at any time by pressing BREAK or CTRL-Y.)

   When you type the option number, an explanation of the option is displayed along with its current setting. For a more detailed description of these options, refer to Appendix A, Server Assigns and Parameters.

   After entering a new value for an option, Install displays again the list of installation options.

3. When all the option values are set correctly, type C and press RETURN.

   Install completes the installation process.

You are now ready to start ViewPoint. For information on starting ViewPoint, see "Starting and Stopping a ViewPoint Pathway" in the next subsection "NonStop™ TS/MP Configuration."

Note. Every time you use the Install routine, ViewPoint installs new requesters, leaving any existing requesters in the library. As a result, the library can become very large, consuming disk space. The Install routine automatically deletes all but the most recent version of each object and compresses the library.

NonStop™ TS/MP Configuration

The Install routine lets you set the most commonly used parameters appearing in the PATHDEFS and PATHCONF configuration files. After installing ViewPoint, you may need to change other default parameter values that do not apply to your system (see Appendix A, Server Assigns and Parametersfor examples). Be sure to review these files for accuracy before starting ViewPoint. To change these parameter values, see "Changing ViewPoint Configuration."

Configuration Procedures

The two configuration files, PATHCONF and PATHDEFS, can be used in one of two ways: either to merge ViewPoint into an existing NonStop™ TS/MP system or to create a separate new NonStop™ TS/MP system. These two procedures are described below.

Adding ViewPoint to an Existing NonStop™ TS/MP System

To merge ViewPoint into an existing NonStop™ TS/MP system, go to the program subvolume where the ViewPoint requesters and servers reside and run PATHCOM using an existing PATHMON name:

```
VOLUME program-subvol
PATHCOM pathmon-name;OBEY PATHDEFS;EXIT
```

Note. If the ViewPoint POBJ code does not already exist in the POBJ application, you must copy it into the POBJ application. See the *NonStop™ TS/MP System Management Manual* for instructions.

Creating a New ViewPoint NonStop™ TS/MP System

To create a separate NonStop™ TS/MP system, you must start the PATHMON for the new NonStop™ TS/MP system on the ViewPoint program subvolume:

```
VOLUME program-subvol
PATHMON /NAME pathmon-name,NOWAIT,CPU pathmon-cpu/
PATHCOM pathmon-name;OBEY PATHCONF;EXIT
```

The default PATHMON name for ViewPoint is $ZVPT.

Starting and Stopping a ViewPoint Pathway

ViewPoint provides routines to start and stop a separate NonStop™ TS/MP system. You always cold start a NonStop™ TS/MP system the first time you run it and again whenever you change the PATHDEFS configuration file. Use cool start to start a NonStop™ TS/MP system if you previously exited ViewPoint to use TACL. The three ViewPoint routines used to start and stop a NonStop™ TS/MP system are:

- To cold start the ViewPoint NonStop™ TS/MP system, type:

```
VOLUME program-subvol
VIEWPT:STARTCOLD [/CPU pathmon-cpu,
PRI process-priority-number/] [pathmon-name]
```

pathmon-cpu

is the PATHMON primary CPU. *cpu* must be different from the back up cpu number or you will receive a PATHCOM error (See error 2009 in Section 14 of the *NonStop™ TS/MP Management Programming Event and Error Message Manual*). *PRI* specifies the priority number of the process. pathmon-name specifies the name of an existing PATHMON. The default name of the ViewPoint PATHMON is $ZVPT.

- To cool start the ViewPoint NonStop™ TS/MP system, type:

```
VOLUME program-subvol
VIEWPT:STARTUP [/CPU pathmon-cpu,
PRI process-priority-number/] [pathmon-name]
```

pathmon-cpu

is the PATHMON primary CPU. *cpu* must be different from the back up cpu number or you will receive a PATHCOM error (See error 2009 in Section 14 of the *NonStop™ TS/MP Management Programming Event and Error Message Manual*). *PRI* specifies the priority number of the process. pathmon-name specifies the name of an existing PATHMON. The default name of the ViewPoint PATHMON is $ZVPT.

- To shut down the ViewPoint NonStop™ TS/MP system, type:

```
VIEWPT:SHUTDOWN [pathmon-name]
```

pathmon-name

is the name of an existing PATHMON process with which ViewPoint is to establish communication. The default name of the ViewPoint PATHMON is $ZVPT. Before shutting down the NonStop™ TS/MP system, this routine makes sure there are no terminals currently running and advises you if there are any running.

The above routines might not be appropriate for all installations. You might want to replace these routines with ones that are specific to your system.

STARTUP Considerations

TACL must be running on the terminal that initiates NonStop™ TS/MP STARTUP. The ViewPoint NonStop™ TS/MP application runs under the process access ID of the user who executes viewpt:startup. The terminal on which viewpt:startup is executed becomes the home terminal of the ViewPoint NonStop™ TS/MP processes.

The Event Management Service (EMS) forwarding distributor processes are not started automatically. You can include them in the ViewPoint NonStop™ TS/MP application. If included, these processes are started by the viewpt:startup routine.

The processes of the Distributed Name Service (DNS) are not started automatically by when you start up ViewPoint. These processes, if applicable to your system, should be started independently.

VIEWPT:STARTUP detects and issues error messages for the following situations:

- The ViewPoint application is already running.
- ViewPoint has not been installed.
- The user does not have access authority to the startup definition files.

Default Configuration Files

The configuration of primary-events and alternate-events displays is kept in an event configuration file. Several such files can exist. ViewPoint provides a default configuration file, EVNTDFLT, that always exists on the ViewPoint data subvolume. EVNTDFLT is created during ViewPoint installation, if it does not already exist at installation time. You can create your own default event configuration file, ZZVPEVNT, on your default logon subvolume. If ZZVPEVNT exists, ViewPoint uses it as the default event configuration file instead of EVNTDFLT.

Status display configuration is kept in a status configuration file. Again, several such files can exist. By default, the file is STATDFLT, which always exists on the ViewPoint data subvolume. STATDFLT is created during ViewPoint installation, if it does not already exist at installation time. You can create your own default status configuration file, ZZVPSTAT, on your default logon subvolume. If ZZVPSTAT exists, ViewPoint uses it as the default status configuration file instead of STATDFLT.

When ViewPoint runs without allowing access to TACL, any ZZVPEVNT or ZZVPSTAT file can exist on the PATHMON user ID default logon subvolume.

Changing ViewPoint Configuration

There are two means available for changing your ViewPoint configuration: Install/CONFIGURE/ and PATHDEFS (the NonStop™ TS/MP configuration file). Note that when using Install/CONFIGURE/ to configure ViewPoint, you can only change those parameters that are modifiable through Install. Thus, if you want to modify other parameters, you must edit the PATHDEFS file.

PATHDEFS defines the ViewPoint configuration. To reconfigure your ViewPoint application, change this file. Some typical situations in which you must reconfigure ViewPoint are described below. (Refer to Appendix A, Server Assigns and Parameters for examples of these situations.)

- Changing existing configuration attributes such as the primary or backup CPU for your application.
- Changing server attributes by adding new server assigns or parameters or by changing existing ones.

- Adding a new custom status server to the application.

To make such changes, you must stop ViewPoint, edit the PATHDEFS file, and then restart ViewPoint from a cold state. When you cold start ViewPoint, it uses the changed PATHDEFS file with its new configuration attributes. You should print a hardcopy of the default PATHDEFS file supplied by ViewPoint before you edit PATHDEFS; you can then edit the file with more confidence. The file is stored on your default logon subvolume.

In some cases, the PATHDEFS file does not have an attribute setting you can change. Many default attributes, such as the home terminal name, do not appear in the ViewPoint PATHDEFS file. In this case, you must add a command. The command could be a Pathway/TS SET TCP or SET PROGRAM, or a NonStop™ TS/MP SET SERVER ASSIGN, or SET SERVER PARAM command.

Accessing Alternate TACL Segment Files

The VIEWPT command requires access to the TACL ZSPI definitions for SPI and NonStop™ TS/MP. (Refer to the *NonStop™ TS/MP System Management Guide*.) ViewPoint assumes that the files containing these definitions are in the default subvolume $SYSTEM.ZSPISEGF for your site. If the files reside in a different subvolume, you should ask your site administrator for the current default subvolume.

When you first use ViewPoint, the definitions are loaded into your private TACL segment and your #USELIST is altered to contain the directory that holds the SPI and NonStop™ TS/MP definitions. If another product uses the same definitions, both products share the definitions.

TACL must be running on the terminal that initiates startup. The ViewPoint NonStop™ TS/MP application runs under the process access ID of the user that invokes the command VIEWPT:STARTUP. The terminal on which VIEWPT:STARTUP is executed becomes the home terminal of the ViewPoint NonStop™ TS/MP processes.

Adding New Server Attributes

When a command for a particular server attribute does not exist in the PATHDEFS file, you must add SET SERVER ASSIGN or SET SERVER PARAM commands to specify server assigns and server parameters, respectively. (Refer to Appendix A, “Server Assigns and Parameters,” for a complete list of the assigns and parameters allowed for ViewPoint servers.)

For example, suppose you want to place your EVENTCX file (containing custom event text) on a different subvolume than your default program subvolume. To do this, you add the CUSTOM-DETAIL assign to the attributes of the ZVPT-EVNT-DETL server. The assign specifies where EVENTCX is located, as shown:

```
SET SERVER ASSIGN CUSTOM-DETAIL, $MYVOL.MYSUBV.EVENTCX
```

As an example of adding a server parameter, you might want to configure the ZVPT-EXTRAS-DELETE program unit into your ViewPoint application. To do this,

you add the CALL-EXTRAS-DELETE parameter to the attributes of the ZVPT-EVNT-COLL server, as shown:

```
SET SERVER PARAM CALL-EXTRAS-DELETE "TRUE"
```

Before you call either the ZVPT-EXTRAS or ZVPT-EXTRAS-DELETE program units, you must be sure they are compiled and available in your object library. You can use SCREEN COBOL Utility Program (SCUP) to check the contents of this library.

Adding a Custom Server

There are several situations in which you might want to add a custom server to your ViewPoint application, for example, a custom status server or any server associated with a ZVPT-EXTRAS program. (Refer to Section 6, Customizing ViewPoint for a discussion of Extras screens and custom status servers.)

You add a custom server to your ViewPoint application by adding all the necessary commands to your PATHDEFS file. For example, adding the following commands to the PATHDEFS file that is supplied with ViewPoint achieves two results: defines the attributes of the custom server, ZVPT-USTA-DISPLAY, described in Appendix D, Sample Custom Status Server and adds the custom server to the ViewPoint application.

```
   RESET SERVER
   SET SERVER AUTORESTART 5
   SET SERVER CPUS (1:0)
   SET SERVER CREATEDELAY 1 SECS
   SET SERVER DEBUG OFF
   SET SERVER DELETEDELAY 10 MINS
[  default HOMETERM ]
   SET SERVER LINKDEPTH 1
   SET SERVER LINKOPENS 1
   SET SERVER MAXLINKS 2
   SET SERVER MAXSERVERS 1
   SET SERVER NUMSTATIC 1
[  default OUT ]
[  default PRIORITY ]
   SET SERVER PROGRAM $VOL.SUBVOL.VPTSTAT
   SET SERVER TMF OFF
   SET SERVER VOLUME $VOL.SUBVOL
ADD SERVER ZVPT-USTA-DISP
```

Remember, for each of these results, you must stop ViewPoint and then cold start it again to run ViewPoint with the changed configuration. You can use the PATHCOM ALTER command to make changes to a running ViewPoint application. To do this, you first freeze and then stop the TCP, program unit, or server you want to alter; make the change; and then thaw and restart the TCP, program, or server. Changes made this way are not, however, retained if you stop and then cold start ViewPoint. To make changes permanent, you must change the PATHDEFS file.

Note. If PATHTCP2 runs at a high PIN, you must make sure that your custom status servers support high-PIN openers.

Running ViewPoint With TACL

The following is a description of how to run ViewPoint with TACL.

Note. If your ViewPoint is to run without access to TACL, skip this discussion and read "Running ViewPoint Without TACL" later in this section.

Once the ViewPoint application is installed and configured, individual terminals gain access to ViewPoint by executing the TACL routine, VIEWPT:

```
VIEWPT [pathmon-name]
```

The default *pathmon-name* is $ZVPT.

Note. The TACL process for the terminal must be a named process.

When ViewPoint is successfully started, any existing function-key definitions are saved and the ViewPoint TACL screen definitions are made available. The ViewPoint product banner is displayed, and the ViewPoint conversational function-key definitions are displayed. (If ViewPoint is set up to run with DSM/PM, the ViewPoint product banner displays the DSM/PM title.) The TACL process access ID is not changed.

A customizing TACL macro file, VPTCSTM on the installed ViewPoint data subvolume, is used if it exists. It is not provided as a part of the ViewPoint product; you must add it if you want to customize the TACL function keys.

The ViewPoint routines detect and issue error messages for the following situations:

- ViewPoint is already initialized on the terminal. The ViewPoint function-key definitions are displayed again.
- The ViewPoint application is not running.
- ViewPoint is requested at a terminal that is not a Compaq 6530-compatible.
- The user does not have access authority (see "Security Considerations" later in this section).
- The initialization steps fail due to NonStop™ TS/MP or communication errors; a specific error message is displayed.

The Color Monitor

ViewPoint can use colors to highlight action or critical events when ViewPoint is running on a Compaq 6AX (or compatible terminal) that is equipped to display colors. The PC6530 emulator (PCT), version C00 or later, must be used. The color-enhanced ViewPoint can be requested by including COLOR as a parameter when you run ViewPoint:

To start ViewPoint with color on a color monitor, at the TACL prompt, type:

```
viewpt /COLOR/ [pathmon-name]
```

pathmon-name

The name that you or your system manager has given the PATHMON. The default is ZVPT.

If the COLOR option is requested, ViewPoint loads a color map into the PCT process during initialization and, if TACL is available, each time the user reenters block mode from the TACL screen. When the user exits from ViewPoint, the color map is reset to the default setting (PCT.INI). (PCT.INI is a file that contains all configurable parameters for the Compaq 6AX or compatible terminal, including colors.)

The default ViewPoint colors are the following:

- White letters on a blue background as the basic colors. You can change this color combination using TACL variables.
- Red letters for critical events.
- Green letters for action events.

If you prefer a different foreground/background color combination, use TACL to #PUSH and #SET one or both of the following TACL variables:

```
:UTILS_GLOBALS:ViewPoint:_FOREGROUND foreground-color
:UTILS_GLOBALS:ViewPoint:_BACKGROUND background-color
```

The color choices for either foreground or background color are BLACK, BLUE, GREEN, CYAN, RED, MAGENTA, BROWN, and WHITE.

Logon Considerations

If the ViewPoint application has access to TACL, the operator must be logged on to TACL before running the VIEWPT routine. After entering ViewPoint, the operator can change the user ID of the TACL process by using LOGON on the TACL conversational screen.

Typing LOGOFF forces an exit from ViewPoint and termination of processes started by the Define Process facility. It also logs you off from TACL.

Processes that are started as a part of the Define Process facility inherit the user ID of the TACL process at the time they are started.

Processes that run in the ViewPoint Pathway application, such as servers and TCPs, run under the user ID of the operator who executes the VIEWPT:STARTUP routine.

Define Process commands can be executed from TACL by entering the commands as described in Section 4, “Process Definition Commands.” These commands are defined in the DP TACL directory (:UTILS:DP). The DP directory name can be put in the TACL #USELIST file so that the command names (DP, PINFO, and so on) can be recognized without prefixing each command with the directory name. The TACLCSTM file on the operator’s default subvolume could contain a TACL command to put the directory name on the #USELIST. For example:

```
USE :UTILS:DP
```

If ViewPoint is run without access to TACL (see "Running ViewPoint without TACL" later in this section), then users do not have access to the background processes available through Define Process commands.

Exiting From ViewPoint

If the ViewPoint application has access to TACL, the operator can exit ViewPoint from any of the base-level screens by pressing the exit key, SF16. The previously saved TACL function-key definitions are then restored, the link from TACL to ViewPoint is broken, and the TACL conversational screen is restored. Any defined processes remain defined.

Note. When you press SF16, ViewPoint continues to run, so you can log back into ViewPoint using the ViewPoint [*pathmon-name*] command. (See the subsection earlier in this section, "Running ViewPoint with TACL.")

If ViewPoint is running on a dedicated terminal (without access to TACL), pressing SF16 stops the ViewPoint application. If ViewPoint was called from another Pathway application, SF16 returns to the calling application.

Running ViewPoint Without TACL

As delivered, ViewPoint allows function-key access to TACL from the terminal. When ViewPoint is installed, however, the application manager can decide to run ViewPoint from a terminal without using TACL. This allows ViewPoint to be called from another Pathway application, makes it possible to provide a security front end to ViewPoint, and allows ViewPoint to run on dedicated terminals.

There are two ways to run the ViewPoint application without access to TACL:

- ViewPoint can be called from an existing Pathway application.
- ViewPoint can be run on a dedicated terminal.

The method for calling ViewPoint from another Pathway application is for the Pathway application to issue a call to ZVPT-MAIN; the call to ZVPT-MAIN starts the ViewPoint application.

The method for starting ViewPoint at a dedicated terminal depends on whether there is a TACL running at that terminal. If there is a TACL running, you can enter the following series of commands:

```
PATHCOM pathmon-name; RUN ZVIEWPOINT; EXIT
```

The PATHMON name must identify an existing ViewPoint PATHMON; by default, the ViewPoint PATHMON name is $ZVPT. These PATHCOM commands start ViewPoint at the terminal where the commands are entered.

If there is no TACL running on the terminal where you want ViewPoint to run, you can define ZVPT-MAIN as the initial program and define and start the ViewPoint terminal from a terminal where TACL is running (usually the operator console). For each

terminal where you want ViewPoint to run, the Pathway configuration might include the following PATHCOM commands:

```
SET TERM INITIAL ZVPT-MAIN
SET TERM TCP ZVPT-TCP
SET TERM FILE network-file-name-of-terminal
SET TERM AUTORESTART 5    <--or whatever value is appropriate
SET TERM BREAK ON         <--if the BREAK key is to be disabled
SET TERM TYPE T16-6530:0
SET TERM PRINTER print-device-name
SET TERM MAXINPUTMSGS 1
ADD TERM logical-terminal-name   <--Pathway name of terminal
START TERM logical-terminal-name
```

The following commands terminate a ViewPoint application running on a dedicated ViewPoint terminal:

```
STOP TERM terminal-name
DELETE TERM terminal-name
```

Refer to the *NonStop™ TS/MP System Management Reference Guide* for detailed information about these commands.

When ViewPoint is not initiated from TACL, the ViewPoint data subvolume is used as the operator logon subvolume to find the EVNTDFLT and STATDFLT files. If default files ZZVPEVNT and ZZVPSTAT exist, they are on the PATHMON user ID default logon subvolume; these files are always used in preference to the EVNTDFLT and STATDFLT files when they exist.

Restrictions When TACL is Not Present

Certain restrictions apply when TACL is not present and integrated with ViewPoint:

- Subsystem commands cannot be issued from within ViewPoint.
- Although the Clip function key can be used to add lines to the Clipboard, its contents are not available from ViewPoint screens.
- File I/O for clipping and printing is performed using the PATHMON user ID instead of the current user ID.
- Screens cannot be printed to a disk file, only to a terminal, process, or printer (this is a Pathway restriction).
- Pressing the TACL function key causes an error message to be displayed.
- The COLOR option is unavailable because TACL provides the color map.
- VPTCSTM is not invoked and cannot be used as a security mechanism.

Security Considerations

ViewPoint implements no new security checking but relies on the checking already provided by the subsystems it uses. Users who install ViewPoint must assign security values to achieve the desired protection.

There are three aspects of ViewPoint security. It controls who can do the following:

- Install and start ViewPoint
- Run ViewPoint
- Perform specific functions through defined processes

Installing and Starting Up ViewPoint

When the system Install program is running, the ViewPoint ISV files are restored with the operator's current default security and ownership values. To install ViewPoint using the ViewPoint Install routine, the installer must have read access to these ISV files.

After ViewPoint is installed on your system, the program and data files that were created during the installation are owned by, and have the security of, the person who installed the application. You should check the security attributes of these files to make sure they are correct for the person planning to start ViewPoint. Anyone starting ViewPoint must have the following combination of read and execute access to the files shown in Table 7-3.

Table 7-2. Files and Their Security Attributes

File Name	Access Required	File Function
$*volume.data-subvol.*TERMTACL	Write	File used during ViewPoint startup to provide mapping from the physical terminal to the TACL server port name
$*volume.subvolume*.ZZEVnnn	Read	EMS event log file
$SYSTEM.SYS*nn*.TEMPLATE	Read	EMS template file
$SYSTEM.SYS*nn*.MEASFH	Read, Execute	Measure program
$SYSTEM.SYS*nn*.TMFSERVE	Read, Execute	TMF program
$SYSTEM.SYS*nn*.SCP	Read, Execute	SCP program
$SYSTEM.SYS*nn*.EMSDIST	Read, Execute	EMS distributor program
$*volume.subvolume.Filter name*	Read	Filter file
$*volume.program-subvol*	Read, Execute	ViewPoint program files
$*volume.data-subvol*	Read	ViewPoint data files
$SYSTEM.SYSTEM.TACLSEGF	Read	TACL library file that contains TACL functions
$SYSTEM.SYSTEM.PATHCOM	Read, Execute	Pathway program files
$SYSTEM.SYSTEM.PATHTCP2	Read, Execute	Pathway program files
$SYSTEM.SYSTEM.PATHTCPL	Read, Execute	Pathway program files
$SYSTEM.SYSTEM.PATHMON	Read, Execute	Pathway program files

In addition, anyone starting up ViewPoint must have read or write access to the collector process $0 (or any EMS Alternate Collector which might be configured) and must have read access to PATHDEFS, PATHCONF, and PATHSTRT. Security settings in PATHDEFS and PATHCONF should be checked and changed, if necessary, so that they reflect the person starting up ViewPoint. You can change these security settings by running the Install routine and changing the values for the following security options: Pathway-OWNER, Pathway-SECURITY, and PROGRAM-SECURITY. (Remember to specify the CONFIGURE option when you first run Install, so that the program changes these option values without also installing new program and data files.)

Pathway-SECURITY should be restricted to the user or group who administers the NonStop™ TS/MP system and must include the user who starts the ViewPoint Pathway system. By default, after installation, it is set to "C" for network community group member.

Note. If the ViewPoint Pathway system is started by a user with the SUPER.SUPER ID, Pathway-SECURITY should be set to "U" or "O".

The user who starts up the ViewPoint applications can also alter the Pathway aspects of the ViewPoint configuration (such as adding, starting, or stopping servers and terminals) while ViewPoint is running. This is done using PATHCOM or the Subsystem Programmatic Interface (SPI) in a Pathway management application.

Running ViewPoint

To run ViewPoint, a user must have write access to the TERMTACL file, which is located on $*volume.data-subvolume* and must have read access to the system TEMPLATE file located on $SYSTEM.SYS*nn* (where *nn* is a 2-digit octal number. The OSIMAGE and many other operating system files reside in this subvolume).

PROGRAM-SECURITY should be set to the user or group who will be running ViewPoint. By default, after installation, it is set to "N" for anyone on the network.

Additionally, anyone planning to run ViewPoint must have read access to an initialization file VPTCSTM (if it exists) in the ViewPoint data subvolume. This file is optional; it consists of a user-written macro that is executed before the ViewPoint application is entered. This macro can implement further security checks. Safeguard can be used to secure this file; you can thus implement requirements for access list-type authorization to ViewPoint.

Users attempting to run the ViewPoint application without proper Pathway program security authorization, see an error message similar to the following:

```
Pathway error 1090 returned from $zvpt.
PATHMON ERROR  - *1090* SECURITY VIOLATION

Initialization failed.
Exiting from ViewPoint.
```

Users attempting to run the ViewPoint application without proper authorization to the VPTCSTM file see an error message similar to the following:

```
VIEWPT
\CITY.$DISCZ.VPTANNA.vptcstm
```

```
                                ^
*ERROR* Security violation

Exiting from ViewPoint.
```

Executing Defined Processes

Define Process macros implement no security. They can be executed by anyone using TACL on a system where the ViewPoint application is installed. The assumption is that processes that are defined implement the desired security. Security checks by utilities are normally based on process access ID; these are still effective—the user starting the process must have the required authorization.

When using Define Process, be aware that defined and running utilities can be inherited by different users who log on to the same terminal. This can be prevented by always stopping all defined processes (PSTOP *) before giving up the terminal.

#LOGOFF has the effect of automatically stopping defined processes and exiting from ViewPoint.

Memory Usage

ViewPoint uses an event cache stored in memory to improve the response time and CPU usage of common event monitoring operations. Storing events in the event cache allows for quick access by the screen display. You can configure the size of the cache so it has enough storage space for your event monitoring purposes. ViewPoint uses these guidelines to compute memory usage for a sixteen-event display page:

- 5 KB of memory for each display page
- 1 MB of memory for 200 display pages

You alter the number of events stored in the event cache with the PRIMARY-CACHESIZE parameter located in your PATHDEFS configuration file. The default value for this parameter is 320 events (20 pages). You can change this value by either editing the PATHDEFS file directly or by running the Install routine to have it updated automatically. For more information on running Install, see “ViewPoint Installation,” the first subsection in this section. For more information on the PRIMARY-CACHE parameter, see Appendix A, Server Assigns and Parameters.

Version Information

The version of the loaded ViewPoint library is contained in the variable T9640D20*vnn*^*ddmmmyy*^VIEWPOINT. The variable contains the version banner, which is printed with the ViewPoint banner during terminal initialization. You can access the directory that contains this variable by typing the following at the prompt:

```
variables :utils:viewpt
```

You can also find the contents of the version variable in the text variable _VIEWPOINTVERSION.

You can text the existence of the version variable within a routine using ViewPoint to ensure that the correct version of the library is loaded, or the contents of _VIEWPOINTVERSION can be tested.

A Server Assigns and Parameters

Server Assigns

Table A-1 is a partial list of the SET SERVER ASSIGN values that are supported by ViewPoint servers. You can use these values to modify your ViewPoint configuration. The default values are shown in parentheses.

You can change server assign values for a cold startup of ViewPoint by changing the Pathway configuration file PATHDEFS. You add new server assigns or replace existing server assigns in the PATHDEFS file and then cold start ViewPoint, as described in Section 7, Installation, Configuration, and Startup.

You use some assigns to supply the same file name to several server classes. When changing file assigns, all occurrences of the assign must be changed. A list of all server classes that implement each assign follows the definition of the assign.

Unqualified file names assume the default volume of the server, which is the ViewPoint data subvolume assigned at installation.

Default server assign values need not be included in PATHDEFS, though you can specify default server assigns for documentation. To specify a nondefault server assign value, you must add a SET SERVER command to the appropriate server class. For example, to assign the EMS log file to a system other than the installation default, add the following command for the ZVPT-EVNT-COLL server class:

```
SET SERVER ASSIGN PRIMARY-COLLECTOR, \NCN.$0
```

Table A-1. Server Assigns *(page 1 of 4)*

ACTION-TAGS	(evntactg)
identifies the file that maps an action tag to (many) event-cache records.	
Servers: ZVPT-EVNT-COLL, ZVPT-GARB-COLL	
ALTERNATE-FILTER	(fltralt)
identifies the compiled filter file that is used as the default filter for the alternate-events displays.	
Servers: ZVPT-EVNT-COLL	
CUSTOM-DETAIL	(eventcx)
identifies the file containing customer-provided text for probable cause and recommended action for events.	
Servers: ZVPT-EVNT-DETL	

Table A-1. Server Assigns *(page 2 of 4)*

DISTRIBUTOR ($system.system.emsdist)

identifies the program file name of the EMS distributor process for the primary-events, alternate-events, or last-events displays. If the default value is used (or defaulted to) for this assign, the distributor is started on a remote system if the collector(s) are all on a single remote system. If a different program file is specified, then it runs on the remote system only if a remote program file name is specified.

Servers: ZVPT-EVNT-COLL, ZVPT-EVLE-COLL

EMS-TEMPLATES (your system template)

identifies the file containing user-written EMS text formatting templates and the facility for testing these templates.

Servers: ZVPT-EVLE-COLL, ZVPT-EVNT-COLL, ZVPT-EVNT-DISP

EVENT-MARKED (evntmrkd)

identifies the file containing event buffers for events selected on an events screen when calling an extras program unit.

Servers: ZVPT-EVNT-DISP, ZVPT-GARB-COLL, ZVPT-EVNT-COLL

EVENT-NOTIFY ($zevn)

identifies the process name of the twenty-fifth notification server (server class: ZVPT-EVNT-NTFY).

Servers: ZVPT-EVNT-COLL

EVENT-TERM (evntterm)

identifies the file that maps the logical terminal and event view onto characteristics of the event display and the events in the cache.

Servers: ZVPT-EVNT-COLL, ZVPT-GARB-COLL

EVENT-TERM-LIST (evnttlst)

identifies the file that maps a logical terminal to the terminal file name (for twenty-fifth line notification) and the TCP name for Pathway unsolicited (UMP) messages.

Servers: ZVPT-EVNT-COLL, ZVPT-GARB-COLL

HELPFILE (helpfile)

identifies the file containing help and error message text.

Servers: ZVPT-HELP-SRVR

LAST-EVENT-CACHE (evntleca)

identifies the file containing all events retained for the last-events display. Events are organized by subject. The number of events retained for each subject is determined by LAST-EVENT-CACHESIZE.

Servers: ZVPT-EVLE-COLL, ZVPT-EVNT-DISP, ZVPT-GARB-COLL

Table A-1. Server Assigns *(page 3 of 4)*

LAST-EVENT-COLLECTOR	($0)

identifies the EMS collector(s) or the log file from which events are read for the last-event display. As many as five collectors or one log file can be selected. Events from all specified collectors are merged. Multiple collectors are specified by creating an edit file containing a list of the collector-process names. The edit file name is given as the value for this assign. The edit file can contain blank lines and comment lines (== indicates the beginning of a comment line). Collector-process names should appear one per line. Each process name can be followed by spaces and a comment (preceded by ==).

Servers: ZVPT-EVLE-COLL

LAST-EVENT-FILTER	(fltrlast)

identifies the compiled filter file that is used for the last-events display.

Servers: ZVPT-EVLE-COLL

LAST-EVENT-SUBJECT	(evntlesb)

identifies the file that provides a count of the events in the LAST-EVENT-CACHE file for each subject.

Servers: ZVPT-EVNT-COLL, ZVPT-EVNT-DISP, ZVPT-GARB-COLL

MEASURE-DATA	(measdata)

identifies the file used by Measure as a measurement data file.

Servers: ZVPT-STAT-COLL

PRIMARY-COLLECTOR	($0)

identifies the EMS collector(s) or the log file from which events are read for the primary-events display. As many as five collectors or one log file can be selected. Events from all specified collectors are merged. Multiple collectors are specified by creating an edit file containing a list of the collector-process names. The edit file name is given as the value for this assign. The edit file can contain blank lines and comment lines (== indicates the beginning of a comment line). Collector-process names should appear one per line. Each process name can be followed by spaces and a comment (preceded by ==).

Servers: ZVPT-EVNT-COLL

PRIMARY-FILTER	(fltrdflt)

identifies the compiled filter file that is used for the primary-events display.

Servers: ZVPT-EVNT-COLL

Table A-1. Server Assigns *(page 4 of 4)*

TANDEM-DETAIL	(eventtd)
identifies the file containing Compaq-provided text for the probable cause and recommended action for events.	
Servers: ZVPT-EVNT-DETL	
TERM-TO-TACL	(termtacl)
identifies the file used during ViewPoint invocation to provide a mapping from the physical terminal to the TACL server port name.	
Servers: ZVPT-TACL-SWCH	
TMF-SERVER	($system.system.tmfserve)
identifies the program file used during status collection to start the server process that provides the TMF transaction rate.	
Servers: ZVPT-STAT-COLL	

Server Parameters

Table A-2 is a partial list of the SET SERVER PARAM values that are supported by ViewPoint servers. You can use these values to modify your ViewPoint configuration. The default values are shown in parentheses.

You can change server parameter values for a cold startup of ViewPoint by changing the Pathway configuration file PATHDEFS. You add new server parameters or replace existing server parameters in the PATHDEFS file and then cold start ViewPoint, as described in Section 7, Installation, Configuration, and Startup.

Default server parameter values need not be included in your PATHDEFS file, though you can, if you choose, add parameters with default values for documentation. To specify a nondefault value for a server parameter, you must add a SET SERVER PARAM command to the appropriate server class. For example, to reposition the EMS event distributor to collect events sent while the ViewPoint application is stopped temporarily, enter the following command for the ZVPT-EVNT-COLL server class:

```
SET SERVER PARAM REPOSITION "TRUE"
```

Table A-2. Server Parameters *(page 1 of 5)*

ALTERNATE-DISTRIBUTOR-CPU (-1)

provides the CPU number in which ViewPoint runs the alternate events distributor. If the CPU field of the Alternate Event Configuration screen is left blank and this parameter is defined with a valid CPU, ViewPoint uses the value of the parameter; otherwise, ViewPoint uses the CPU of the server.

Servers: ZVPT-EVNT-COLL

AUTO-SUSPEND (FALSE)

allows event collection to be suspended if an outstanding critical or action event is about to be scrolled out of the event cache. The operator is notified that an unacknowledged critical or action event is about to be scrolled out of the event cache. The event collection resumes once the operator acknowledges the event. To turn on the option, set this parameter to TRUE.

Servers: ZVPT-EVNT-COLL

BELLRESET (-1)

provides the volume setting (0-7) to which to restore the bell if the bell volume is changed on a write to the twenty-fifth line. By default (-1), the bell volume is not reset. If a number higher than 7 is specified, the number 7 is used, and if a number lower than -1 is specified, -1 is used.

Servers: ZVPT-EVNT-COLL

BELLVOLUME (-1)

provides the volume setting (0-7) of the bell for the 6530 terminal. The bell is used when the ZVPT-EVNT-NTFY server writes to the twenty-fifth line. The default value (-1) indicates the terminal-configured setting is used. If a number higher than 7 is specified, the number 7 is used, and if a number lower than -1 is specified, -1 is used. If 0 is used, the bell is not rung.

Servers: ZVPT-EVNT-COLL

CALL-EXTRAS-DELETE (FALSE)

indicates whether the program unit ZVPT-EXTRAS-DELETE is called whenever a manual delete is attempted on the Primary-Events screen. TRUE indicates the program unit is called; FALSE (the default setting) indicates that the program is not called.

Servers: ZVPT-EVNT-COLL

CLEANUP-HOUR (23)

identifies the hour on a 24-hour clock at which the garbage collection cleanup tasks are scheduled to run. The default value is 23 (11 p.m.). Possible values are 0 through 24. The value 24 is interpreted as 0.

Servers: ZVPT-GARB-COLL

Table A-2. Server Parameters *(page 2 of 5)*

DISPLAY-KANJI (FALSE)

supports displaying special characters, such as Kanji, that are embedded in the event text. If this parameter is set to FALSE or not defined, ViewPoint strips out non-displayable characters that have ASCII code values below 32 (space character) or between 126 (tilde character) and 160 (a graphic character) from the event text. When the parameter is set to TRUE, ViewPoint strips out only characters that have ASCII code values of 127 (delete character) or below 32 (space character). Stripping these characters allows characters such as Kanji to be displayed.

CAUTION: This feature is dependent on the displaying support feature of the terminal. Do not try to set this parameter value to TRUE when the terminal does not support displaying characters like Kanji; otherwise, this could cause the terminal to display garbage characters or even to lock up.

Servers: ZVPT-EVNT-COLL, ZVPT-EVNT-DISP

EVENT-PREFIX (1)

indicates the prefix text that accompanies events displayed on the events screens. Possible values are 1 (default value), 2, and 3. If 1 is used, the prefix shows the time of the event and name of the system generating the event in the format “hh:mm \sysname.” Value 2 shows a prefix with the date and time of the event and system name in the format “mon-dd hh:mm \sysname.” Value 3 shows a prefix with the time of the event, system name, and a character (!, *, or blank) in the beginning of the prefix that indicates the type of event (critical, action, or other). If value 3 is used, the prefix is in the format “^hh:mm \sysname” where ^ is the ! character (for critical events), the * character for action events, and a blank character for all other events.

A unique event prefix can be created by modifying the existing event template used by ViewPoint. See Section 6 for more information on creating a custom event prefix.

Servers: ZVPT-EVNT-COLL, ZVPT-EVNT-DISP

HANGAROUND (FALSE)

indicates whether a process remains alive after the last opener closes it. The default value is FALSE (the process goes away after the last opener closes it). If TRUE, the process never voluntarily terminates (TRUE is the recommended setting).

Servers: ZVPT-EVNT-NTFY

IGNOREALTEVENT (FALSE)

Allows two event collection servers to run simultaneously and thereby improve event reception performance. IGNOREALTEVENT indicates whether the server should ignore the alternate-events display. If the display is ignored, no events are collected for that display and any requests for service are rejected with a ViewPoint error 325. The default value is FALSE, meaning that all displays are managed from one server.

Note that this feature is used in the default Pathway configuration to create two servers: ZVPT-EVNT-COLL, which manages the primary and alternate event displays, and ZVPT-EVLE-COLL, which manages the last events display.

Servers ZVPT-EVLE-COLL

Table A-2. Server Parameters *(page 3 of 5)*

IGNORELASTEVENT (FALSE)

allows two event collection servers to run simultaneously and thereby improve event reception performance. IGNORELASTEVENT indicates whether the server should ignore the last-event display. If the display is ignored, no events are collected for that display and any requests for service are rejected with a ViewPoint error 325. The default value is FALSE, meaning that all displays are managed from one server.

Note that this feature is used in the default Pathway configuration to create two servers: ZVPT-EVNT-COLL, which manages the primary and alternate event displays, and ZVPT-EVLE-COLL, which manages the last- events display.

Servers: ZVPT-EVNT-COLL

IGNOREPRIEVENT (FALSE)

allows two event collection servers to run simultaneously and thereby improve event reception performance. IGNOREPRIEVENT indicates whether the server should ignore the primary-events display. If the display is ignored, no events are collected for that display and any requests for service are rejected with a ViewPoint error 325. The default value is FALSE; that is, all displays are managed from one server.

Note that this feature is used in the default Pathway configuration to create two servers: ZVPT-EVNT-COLL, which manages the primary-events and alternate-events displays, and ZVPT-EVLE-COLL, which manages the last events display.

Servers: ZVPT-EVLE-COLL

LAST-DISTRIBUTOR-CPU (-1)

provides the cpu number in which ViewPoint runs the last events distributor. If this parameter is defined with a valid CPU number, ViewPoint uses the value of the parameter; otherwise, ViewPoint uses the CPU of the server.

Servers: ZVPT-EVLE-COLL

LAST-EVENT-CACHESIZE (10)

provides the number of events retained in the last-events cache for each subject. Newer events overwrite the oldest for each subject. You can specify a value of 1 through 32767. If you specify a value less than 1, the value 1 is used.

The value is used each time a new event subject is logged to the last-events database. A new LAST-EVENT-CACHESIZE value does not take effect for subjects already in the database. Most subjects eventually expire and are deleted by the ZVPT-GARB-COLL server, but frequently used subjects might never be deleted.

You might want to clean out the database when changing the cache size so that all subjects use the new cache size. You can do this with the FUP command PURGEDATA (EVNTLESB, EVNTLECA) with the files closed.

Servers: ZVPT-EVLE-COLL

LAST-EVENT-MAX-HOURS (72)

provides the maximum age that events in the last-events cache are allowed to attain before they are deleted by the garbage-collection cleanup task. Possible values are 0 through 32767. The cleanup task is scheduled to run at the time specified by the parameter CLEANUP-HOUR.

Servers: ZVPT-GARB-COLL

Table A-2. Server Parameters *(page 4 of 5)*

MAXDELAY (1440)

identifies the maximum time in minutes that ViewPoint allows the I/O of a message to a terminal's 25th line to be pending until the I/O completes. The default is 1440 minutes (one day). If a message does not complete in this amount of time, ViewPoint cancels the I/O.

Servers: ZVPT-EVNT-NTFY

MAXBUFFERS (50)

provides the maximum number of messages that can be pending against all devices. The default value is 50. If ViewPoint needs to send another message and there are already 50 outstanding messages, the oldest outstanding message is canceled. Do not change this value; altering MAXBUFFERS can have unexpected side effects.

Servers: ZVPT-EVNT-NTFY

MAXQUEUE (5)

provides the maximum number of messages that can be pending against one device at the same time. If another message must be sent to a device and there are already 50 messages pending, the oldest message pending against the device is canceled. Possible values are 1 through 14. If a number greater than 14 is specified, 14 is used. The server default value is 5; however, the ViewPoint standard installed configuration value is 2. The value 2 is the recommended value.

Servers: ZVPT-EVNT-NTFY

MAXTERMS (20)

provides the maximum number of devices that can be open with pending twenty-fifth line messages at one time. If another device must be opened and the maximum are already open, all messages pending against the device that has the oldest outstanding message are canceled and that device is closed. Do not change this parameter; altering MAXTERMS can have unexpected side effects.

Servers: ZVPT-EVNT-NTFY

NOTIFY-ALT-EVENT (FALSE)

allows the status line to be updated and the bell to sound for critical and action events arriving for the Alternate Events screens. When the parameter is set to TRUE, ViewPoint selects the Bell and Status Line option for Alternate Events. If the parameter is not defined or it is set to FALSE, ViewPoint selects the Bell and Status Line for Primary Events. When the Bell and Status Line option is set for the Alternate Events display, each terminal has its own private Alternate Events Status Line. The Bell and Status Line for a terminal is activated when the Alternate Events display is brought up for the first time.

Servers: ZVPT-EVNT-COLL

PRIMARY-CACHESIZE (320)

provides the number of events retained in the event cache for the primary view of events. Sixteen events are displayed on each page of the display; thus the default value of 320 events is 20 pages. Possible values for this parameter are 1 through 12000. If you specify a value less than 1, ViewPoint assumes the cache size is 1. If a value higher than 12000 is specified, ViewPoint sets the primary cache size to 12000.

Servers: ZVPT-EVNT-COLL

Table A-2. Server Parameters *(page 5 of 5)*

PRIMARY-DISTRIBUTOR-CPU (-1)

provides the CPU number for the CPU in which ViewPoint runs the primary events distributor. If this parameter is defined with a valid CPU, ViewPoint uses the value of the parameter; otherwise, ViewPoint uses the CPU of the server.

Servers: ZVPT-EVNT-COLL

RECOVER-CACHE (FALSE)

indicates whether primary-event and alternate-event distributors should be repositioned to the oldest event in the cache when ZVPT-EVNT-COLL starts the distributors. The ViewPoint event collector examines this flag when it is restarted after a CPU failure or shutdown. The flag indicates how distributors should be positioned when they are restarted by the ViewPoint event collector. FALSE means that the distributor is not repositioned. In this case, the newly created EMS distributor returns only events that are generated after it was started. TRUE means that the distributor is repositioned; that is, the new distributor returns events starting at the oldest event in the cache before the previous distributor stopped. The default value is FALSE and is the recommended setting for best performance. Note that this flag does not affect the last-events distributor. (Note that the REPOSITION and RECOVER-CACHE parameter are the same, and can be used interchangeably.)

Servers: ZVPT-EVNT-COLL

REPOSITION (FALSE)

This parameter is the same as RECOVER-CACHE (see the preceding description).

Servers: ZVPT-EVNT-COLL

SAMPLE-AGE-MAX-MINUTES (60)

provides the maximum age in minutes that status samples can attain before they are considered obsolete for display by status display. This age is used by the garbage-collection server to delete obsolete samples. Possible values are 1 through 32767.

Servers: ZVPT-STAT-DISP, ZVPT-GARB-COLL, ZVPT-STAT-CNFG

STOPDELAY (1)

provides the maximum number of minutes to wait for messages to complete after the ViewPoint last opener closes a process before terminating the process. The default value is 1. This parameter is relevant only if HANGAROUND is FALSE. Setting STOPDELAY to 0 causes the process to terminate immediately after the last opener closes it. Possible values are 0 through 32767.

Servers: ZVPT-EVNT-NTFY

SUPPRESSUNANSWERED (FALSE)

indicates whether to suppress the message that notifies each ViewPoint terminal when an outstanding event rolls off the end of the primary-events cache. The default value is FALSE; that is, the message is not suppressed. When set to TRUE, the message is suppressed.

Servers: ZVPT-EVNT-COLL

B Error Messages

This appendix gives a list of the error and advisory messages that are displayed by ViewPoint. (An advisory message, in contrast to an error message, advises the operator of successful completion of an operation or tells the operator what step to perform next.)

Each error or advisory message has an associated message number. However, these numbers are displayed only for error messages; the number is not displayed for advisory messages. Advisory messages are identified by an (A) in this list.

The messages are displayed on line 20 or 21 of the block-mode screens. In the case of error messages, the message number follows the text of the message. Some error messages are displayed with two numbers; a number that provides additional internal error information, such as a file-system error number, can follow the ViewPoint error number.

For example, error messages are displayed in the following formats for errors 6 and 205:

```
Select lines to clip.  Then press F10.            0006
Status collection server send error:              0205   0001
```

Error 6 has only the ViewPoint error number; error 205 has an additional number. This second number is a SCREEN COBOL SEND error. Explanations of errors associated with the additional error numbers can be found in the following manuals:

- File-system error, *Guardian Procedure Errors and Messages Manual*
- SCOBOLX SEND error, *Pathway/TS SCREEN COBOL Reference Manual*
- SCOBOLX CALL error , *Pathway/TS SCREEN COBOL Reference Manual*
- Unsolicited message processing error, *Pathway Application Programming Guide*
- SPI procedure error, *SPI Programming Manual*
- Measure error, *Measure Reference Manual*
- EMS distributor error, *Event Management Service (EMS) Manual*
- EMSTEXT error, *Event Management Service (EMS) Manual*
- NEWPROCESS error, *Guardian Procedure Errors and Messages Manual*

Fatal server error messages are reported to $0 and appear on the console log. These messages provide a P-register trace call location for a process that abnormally ends because of an error. In this case, the first number is the current location. You can use LMAP to identify the procedure and the offset of the error in the program. The following is an example:

```
\MASTER.$Z117 : EVCOLL: error during initialization 0336
\MASTER.$Z117 : P-REG TRACE %006551 %005343 %000232
```

Error

In this example, the first message indicates ViewPoint error number 0336, listed in this appendix as “Please enter a correct filter file name.” The second message is the P-register trace call location for the process.

You should include all numbers provided with the error messages along with any documentation of a problem when you report the problem.

For ease of reference, the error messages in this appendix list the error number first.

0001 (A)

```
Screen printed.
```

Cause. The currently displayed screen has been printed.

Effect. This is for information only; there is nothing you need to do.

Recovery. N/A

0003

```
Type correct destination name.  Then press F9 to print.
```

Cause. When trying to print your screen, you entered an invalid spooler file name.

Effect. The screen is not printed.

Recovery. Enter the correct spooler file name and press F9 to print your screen.

0004 (A)

```
Line clipped.
```

Cause. The line you selected has been copied to the clipboard file ZZVPCLIP in your default logon subvolume.

Effect. Information only

Recovery. N/A

0005 (A)

```
Lines clipped.
```

Cause. The lines you selected have been copied to the clipboard file ZZVPCLIP in your default logon subvolume.

Effect. Information only

Recovery. N/A

0006

```
Select lines to clip.  Then press F10.
```

Cause. You did not select any lines on the screen to copy to your clipboard file.

Effect. No lines are copied to the clipboard file.

Recovery. Select lines on the screen by typing any character in column 1 of those lines and press F10.

0007

```
Couldn't clip to the clipboard file.
```

Cause. ViewPoint could not copy the specified lines to your clipboard file because the file is inaccessible.

Effect. No lines are copied.

Recovery. Check the security setting of your clipboard file to make sure it is accessible.

0008

```
There is no previous page.
```

Cause. You tried to scroll backward when you are already at the first page.

Effect. You cannot display a previous page.

Recovery. Display a later page or try a different function.

0009

```
This is the last page.
```

Cause. You tried to scroll forward when you are already at the last page.

Effect. You cannot display a later page.

Recovery. Display a previous page or try a different function.

0010

```
You pressed an inactive function key.  Choose another.
```

Cause. The key you pressed is inactive on the current screen.

Effect. Your current operation might be unsuccessful.

Recovery. Press F15 for Help in choosing another key.

0012

```
Garbage Collection server send error:
```

Cause. A server SEND error was received while ViewPoint was invoking the garbage-collection server. The termination status of the SEND is displayed as the second number in the error message.

Effect. Garbage collection is not successful.

Recovery. See the *Pathway SCREEN COBOL Reference Manual* for an explanation of SEND errors.

0013

```
Garbage Collection server request error:
```

Cause. A request error was received while ViewPoint was invoking the garbage-collection server. This is an internal error.

Effect. Garbage collection is not successful.

Recovery. Report this internal error to a Compaq representative, and be sure to supply the error numbers.

0014

```
Error occurred while receiving an unsolicited message:
```

Cause. An error occurred while ViewPoint was receiving a Pathway unsolicited message.

Effect. The event screens may not be updated correctly.

Recovery. See the *Pathway Application Programming Guide* for an explanation of the error.

0015

```
Error occurred while replying to an unsolicited message:
```

Cause. An error occurred while replying to a Pathway unsolicited message.

Effect. The event screens may not be updated correctly.

Recovery. See the *Pathway Application Programming Guide* for an explanation of the error.

0019

```
An internal error was returned from SSINIT:
```

Cause. An SSINIT on a buffer resulted in an error. The error number is included in the message.

Effect. Your current operation may be unsuccessful.

Recovery. See the *SPI Programming Manual* for an explanation of the error.

0020

```
An internal error was returned from SSGET:
```

Cause. An SSGET on a buffer resulted in an error. The error number is included in the message.

Effect. Your current operation may be unsuccessful.

Recovery. See the *SPI Programming Manual* for an explanation of the error.

0021

```
An internal error was returned from SSPUT:
```

Cause. An SSPUT on a buffer resulted in an error. The error number is included in the message.

Effect. Your current operation may be unsuccessful.

Recovery. See the *SPI Programming Manual* for an explanation of the error.

0022

```
Error returned from the TCP from an UMP message:
```

Cause. An Unsolicited Message Processing (UMP) message resulted in an error. The Pathway error number is provided as the second number in the message.

Effect. Your current operation may be unsuccessful.

Recovery. See the *Pathway Application Programming Guide* for an explanation of UMP errors.

0023

```
An UMP message was rejected by the TCP:
```

Cause. The Pathway on-error clause was triggered by the call to the EXTRAS program unit. The Pathway error number is provided as the second number in the message.

Effect. Your current operation may be unsuccessful.

Recovery. See the *Pathway Application Programming Guide* for an explanation of the error.

0024

```
A CALL error was returned when invoking an EXTRAS program.
```

Cause. The Pathway on-error clause was triggered by the call to an EXTRAS program unit.

Effect. The Extras screen is not displayed.

Recovery. See the *Pathway SCREEN COBOL Reference Manual* for an explanation of CALL errors.

0025

```
No EXTRAS screen is available.
```

Cause. The Pathway on-error clause was triggered by the call to the EXTRAS program unit. The Pathway error number indicates that the EXTRAS requestor (program unit ZVPT-EXTRAS) is not in the TCLPROG directory.

Effect. The Extras screen is not displayed.

Recovery. Make sure the ZVPT-EXTRAS program unit exists in the TCLPROG directory. For information on creating ZVPT-EXTRAS, see Section 6, “Adding Extras Screens.”

0026

```
No DELETE-EXTRAS program unit is available.
```

Cause. The Pathway on-error clause was triggered by a CALL to the EXTRAS-DELETE program unit. The Pathway error number indicates that the EXTRAS requestor (program unit ZVPT-EXTRAS-DELETE) is not in the TCLPROG directory.

Effect. The Delete-Extras screen is not displayed.

Recovery. Make sure the ZVPT-EXTRAS-DELETE program unit exists in the TCLPROG directory. For information on creating ZVPT-EXTRAS-DELETE, see Section 6, “EXTRAS-DELETE Interface."

0027

```
Enter correct page number.
```

Cause. You entered an incorrect page number in the option line.

Effect. The requested page is not displayed.

Recovery. Enter a correct number that does not include decimal points or letters.

0030

```
A CALL error was returned when invoking the DSM/PM program:
```

Cause. The Pathway on-error clause was triggered by the call to the DSM/PM program unit. The Pathway error number is provided as the second number in the message.

Effect. DSM/PM is not invoked.

Recovery. See the *Pathway SCREEN COBOL Reference Manual* for an explanation of CALL errors.

0031

```
An error was returned by the DSM/PM program:
```

Cause. The DSM/PM program encountered an error.

Effect. Your current operation is unsuccessful.

Recovery. See the *DSM/Problem Manager Installation and Customization Guide* or the *DSM/Problem Manager User Guide* for an explanation of DSM/PM errors.

0100

```
Error in send to TACL switch server:
```

Cause. A Pathway error was reported on the SEND. The termination status is saved in LS-ERROR-SUBSTATUS.

Effect. TACL may not be available.

Recovery. See the *Pathway SCREEN COBOL Reference Manual* for an explanation of SEND errors.

0101

```
Error in request to the TACL switch server:
```

Cause. This is an internal error.

Effect. Your current operation may be unsuccessful.

Recovery. Supply the error numbers when reporting this internal error to a Compaq representative.

0102

```
ZVPT-TACL-WORK call error:
```

Cause. A Pathway error was reported on the CALL.

Effect. Your current operation may be unsuccessful.

Recovery. See the *Pathway SCREEN COBOL Reference Manual* for an explanation of CALL errors.

0103

```
Error opening the terminal-to-TACL file:
```

Cause. An error occurred while ViewPoint was opening the TERMTACL file.

Effect. The initialization of ViewPoint is aborted.

Recovery. Try recreating the data files by typing the following command: FUP /IN FUPCMDS/.

0104

```
Error reading the terminal-to-TACL file:
```

Cause. An error occurred while ViewPoint was reading the TERMTACL file. TERMTACL may be bad.

Effect. The initialization of ViewPoint is aborted.

Recovery. Try recreating the data files by typing the following command: FUP /IN FUPCMDS/.

0105

```
Error occurred on TACL communication file:
```

Cause. An error occurred during a WRITEREAD on the TACL #SERVER file. If TACL sends an error besides FEPATHDOWN or FEDEVDOWN, this error occurs.

Effect. Your current operation may be unsuccessful.

Recovery. Supply the error numbers when reporting this internal error to a Compaq representative.

0106

```
Reply from TACL was too small.
```

Cause. The message received from TACL was too small.

Effect. Your current operation may be unsuccessful.

Recovery. Make sure the TACL library version matches the ZVPT-TACL-SWCH version. Supply the error numbers when reporting this internal error to a Compaq representative.

0107

```
Error writing to the terminal-to-TACL file:
```

Cause. An error occurred while writing to the TERMTACL file.

Effect. Initialization is aborted.

Recovery. Try recreating the data files by typing the following command: FUP /IN FUPCMDS/.

0108

```
TACL is unavailable.
```

Cause. No link has been set up between TACL and Pathway. TACL must be running on the terminal that initiates the Pathway startup.

Effect. It is not possible to switch to TACL from this environment.

Recovery. Set up a Pathway link. See Section 7, “Running ViewPoint Without TACL,” for information on running ViewPoint when TACL is not present.

0203

```
ZVPT-STAT-CNFG SEND failure:
```

Cause. A Pathway error was reported during a send operation.

Effect. The current status configuration file is not updated.

Recovery. See the *Pathway SCREEN COBOL Reference Manual* for an explanation of SEND errors.

0204

```
ZVPT-STAT-CNFG request error:
```

Cause. A request error was received while ViewPoint was invoking the status configuration server.

Effect. The current status configuration file is not updated.

Recovery. Report this internal error to a Compaq representative, and be sure to supply the error numbers.

0205

```
Status collection server send error:
```

Cause. An error was returned from the SEND to the collection server. The SEND error returned is included in the message.

Effect. Status collection may not be successful.

Recovery. See the *Pathway SCREEN COBOL Reference Manual* for an explanation of SEND errors.

0206

```
Status collection server request error.
```

Cause. A request to the status collection server resulted in an error. This is an internal error. The server does not recognize the request.

Effect. Status collection may not be successful.

Recovery. Supply the error numbers when reporting this internal error to a Compaq representative.

0207

```
Please type a display type for this server, or use Scan.
```

Cause. You did not enter a valid display type for the specified status server in the Display Type field.

Effect. No display type is selected.

Recovery. Enter a valid display type or press F13 (Scan) to fill in the Display Type field with a default selection.

0208

```
The object specified is not valid for the status type.
```

Cause. The object specified in the Object Name field is not valid for the status type specified.

Effect. No object name is selected.

Recovery. Enter a valid object name and press F12.

0210

```
The parameters specified in the item configuration are
incompatible.
```

Cause. The configuration values given (aside from status type and object) are incompatible.

Effect. No values are selected.

Recovery. Enter new configuration values and press F12.

0211

```
The object of this status display is unavailable.
```

Cause. The object of the status display is not available because the monitor or resource is down.

Effect. The object is not displayed.

Recovery. Wait until the object is available before trying to display it.

0212

```
Use Scan key to retrieve an item before pressing Scan Next.
```

Cause. The Scan Next function key (SF13) should not be pressed until a status item has been retrieved using the Scan function key (F13).

Effect. No new status items are retrieved.

Recovery. Press F13 to retrieve a new status item, then press SF13.

0213

```
Provide PATHMON name on the specified system.
```

Cause. You entered a system name without a PATHMON name in the Under PATHMON field on the Status Item Configuration screen.

Effect. No PATHMON name is selected.

Recovery. Enter a correct PATHMON name. If the current (NCN) PATHMON is desired, then both system and PATHMON names should be blank.

0214 (A)

```
Item deleted from configuration.
```

Cause. A status item was successfully deleted from the Network Status Summary screen.

Effect. Information only

Recovery. N/A

0215 (A)

```
Item added to configuration.
```

Cause. A status item was successfully added to the Network Status Summary screen.

Effect. Information only

Recovery. N/A

0216 (A)

```
Item changed in configuration.
```

Cause. A status item was successfully changed in the Network Status Summary screen.

Effect. Information only

Recovery. N/A

0217

```
An error occurred while invoking an internal status program
unit.
```

Cause. The on-error exit was taken from a CALL to a status configuration program unit.

Effect. Status collection may not be successful.

Recovery. Termination status should be examined to determine the cause of the error.

0218

```
Please select items and press F6 to delete.
```

Cause. The Delete function key (F6) was pressed when no items were selected.

Effect. No items are deleted.

Recovery. Select an item by positioning the cursor next to it and press F6.

0219 (A)

```
Items deleted from configuration.
```

Cause. Items have been deleted from the status item configuration.

Effect. Information only

Recovery. N/A

0220

```
MEASURE subsystem is not running on the specified system.
```

Cause. The Measure subsystem is not running on the system.

Effect. Objects normally monitored by Measure may display an error or invalid measurement.

Recovery. Start Measure by entering the START MEASSUBSYS command in MEASCOM.

0221

```
MEASURE error returned while starting the measurement:
```

Cause. The Measure subsystem is running, but an error was returned during configuration and startup of the measurement.

Effect. Objects normally monitored by Measure may display an error or invalid measurement.

Recovery. See the *Measure Reference Manual* for an explanation of Measure errors.

0222

```
MEASURE error returned while sampling the object:
```

Cause. A Measure error was returned by the MEASREADACTIVE call for the specified object.

Effect. An ILLEGALMEASNUM error results in an automatic restart of the measurement.

Recovery. See the *Measure Reference Manual* for an explanation of Measure errors.

0223

```
File error reported while reading status configuration.
```

Cause. A file-system error was reported while ViewPoint was opening or reading the status configuration file. The error reported is shown as the second number in the message.

Effect. Certain status items may be missing from your screen.

Recovery. See the *System Procedure Errors and Messages Manual* for an explanation of file-system errors.

0224

```
The status configuration file has an incorrect file code.
```

Cause. An incorrect file code was specified for the status configuration file.

Effect. Certain status items may be missing from your screen.

Recovery. Enter a correct code. The file code for status configuration files should be 450.

0225

```
File error occurred while updating the configuration file:
```

Cause. A file-system error occurred on the current configuration file. The error number is specified as the second number in the message.

Effect. Certain status items may be missing from your screen.

Recovery. See the *System Procedure Errors and Messages Manual* for an explanation of file-system errors.

0226

```
Provide a correct status configuration file name.
```

Cause. The file name provided could not be converted into a valid internal file name.

Effect. No status configuration file is created.

Recovery. Enter a new file name for your status configuration file.

0227

```
File error occurred during configuration file creation.
```

Cause. A file-system error occurred while creating, opening, or writing to the new configuration file. The file-system error number is provided as the second number in the message.

Effect. No status configuration file is created.

Recovery. See the *System Procedure Errors and Messages Manual* for an explanation of file-system errors.

0228

```
The update interval specified is not valid (1 to 3600).
```

Cause. The interval specified is not in the range of values allowed.

Effect. No interval is selected.

Recovery. Enter a number between 1 and 3600.

0229 (A)

```
Press F14 again to create the new status configuration file.
```

Cause. A confirmation message is displayed after a new file name is entered and F14 is pressed (Profile screen).

Effect. Information only

Recovery. N/A

0230 (A)

```
The status configuration file has been changed.
```

Cause. The status configuration file has been changed. This is reported from the Profile screen when the Configuration File field is modified.

Effect. Information only

Recovery. N/A

0231 (A)

```
The status configuration file has been created.
```

Cause. The status configuration file has been created. This is reported from the Profile screen when the Configuration File field is modified and the confirmation flag is set.

Effect. Information only

Recovery. N/A

0232

```
Device information is unavailable due to internal error
starting PUP.
```

Cause. This error is returned when the server is asked to sample a device before PUP has completed its LISTDEV operation. The requester retries this request four times before reporting this error.

Effect. Certain status items may be missing from the Network Status Summary screen.

Recovery. Contact your system administrator for help with correcting this error.

0233 (A)

```
The update interval is changed in the configuration.
```

Cause. The update interval has changed in the current status configuration file. The change is effective with the next status screen update.

Effect. Information only

Recovery. N/A

0234

```
Please type an existing server class and press F12.
```

Cause. A SEND error was received by the configuration requester when it attempted to send to the specified server class. A status server class must be defined in a ViewPoint Pathway environment (using ADD SERVER) if it is not the standard server class.

Effect. No server class name is selected.

Recovery. Enter a valid server class and press F12.

0235

```
NEWPROCESS of subsystem server results in a file-system
error.
```

Cause. The NEWPROCESS procedure of the TMFSERVE or Measure subsystems report this error if an error is detected on the program file, the library file, the process name, the swap file, or the extended-segment swap file. The file-system error number is supplied.

Effect. Certain status items may not be measured.

Recovery. See the *System Procedure Errors and Messages Manual* for an explanation of NEWPROCESS and file-system errors.

0236

```
NEWPROCESS of subsystem server results in an error:
```

Cause. The NEWPROCESS of TMFSERVE or Measure subsystems reports this error indicating trouble in starting the process. The high-order portion of the NEWPROCESS error is supplied.

Effect. Certain status items may not be measured.

Recovery. See the *System Procedure Errors and Messages Manual* for an explanation of NEWPROCESS errors.

0237

```
File error in sending the subsystem server startup message:
```

Cause. The open and write of the TMFSERVE startup message reports this message. The process is stopped. The file-system error number is included as the second number in the message.

Effect. Certain status items may not be measured.

Recovery. See the *System Procedure Errors and Messages Manual* for an explanation of file-system errors.

0238

```
File error in sending a request to the subsystem server:
```

Cause. The open and write of the TMFSERVE process request report this message. The process is stopped. The file-system error number is included in this message.

Effect. Certain status items may not be measured.

Recovery. See the *System Procedure Errors and Messages Manual* for an explanation of file-system errors.

0239

```
An internal error was detected in the programmatic interface.
```

Cause. An SSINIT, SSGET, SSPUT, or other SPI routine returned a nonzero status. This is an internal error. The SPI error number is included in this message.

Effect. ViewPoint may not report correct information.

Recovery. See the *SPI Programming Manual* for an explanation of SPI procedure errors.

0240

```
The sampled subsystem server return code indicates an error.
```

Cause. A nonzero return code was received from the server. Its value is supplied in the message. The interpretation of this number is subsystem dependent.

Effect. Certain status items may not be measured.

Recovery. Supply the error numbers when reporting this internal error to your Compaq representative.

0241

```
The PATHMON name must be a local or network process name.
```

Cause. The PATHMON name specified was not a proper process name.

Effect. No PATHMON name is selected.

Recovery. Enter a correct PATHMON name in the form [*sysname*.]$*pname*.

0242

```
Status display server send error:
```

Cause. An error was returned from the SEND to the status display server, ZVPT-STAT-DISP. The error returned is included in the message.

Effect. Certain status items may not be measured.

Recovery. See the *Pathway SCREEN COBOL Reference Manual* for an explanation of SEND errors.

0243

```
Status display server request error:
```

Cause. A request to the status display server resulted in an internal error.

Effect. Certain status items may not be measured.

Recovery. Supply the error numbers when reporting this internal error to a Compaq representative.

0244

```
File error occurred while accessing the status cache file:
```

Cause. A file-system error occurred while ViewPoint was accessing the status cache file. The status-display server refers to a file that by default is STATCACH on the default subvolume. This can be changed by the status-cache assign. The file-system error number is provided with this message.

Effect. Certain status items may not be measured.

Recovery. See the *System Procedure Errors and Messages Manual* for an explanation of file-system errors.

0245

```
Sample cannot be computed; interval is less than
sample-age-max.
```

Cause. A status item record was deleted before the percentage busy value could be computed.

Effect. The status item is not measured.

Recovery. Check to see if the screen update interval is longer than the value specified for the SAMPLE-AGE-MAX-MINUTES server parameter.

0246

```
File error returned while starting the measurement:
```

Cause. The Measure subsystem is running, but a file-system error was returned from the MEASOPEN call in starting the measurement. This probably indicates that an error occurred during access to the MEASDATA file on the default subvolume. The file-system error number is included in the message.

Effect. Certain status items may not be measured.

Recovery. See the *System Procedure Errors and Messages Manual* for an explanation of Measure and file-system errors.

0247

```
File error returned from NEWPROCESS while starting the
measurement:
```

Cause. The Measure subsystem is running, but a NEWPROCESS file-system error was returned from the MEASOPEN call in starting the measurement. This probably indicates an error during the start of the Measure file-handling program (MEASFH on SYS*nn* subvolume). The file-system error number is included in the message.

Effect. Certain status items may not be measured.

Recovery. See the *System Procedure Errors and Messages Manual* for an explanation of Measure and file-system errors.

0248

```
NEWPROCESS error occurred while starting the measurement:
```

Cause. The Measure subsystem is running, but the MEASOPEN call returned a NEWPROCESS error when starting the measurement. This probably indicates an error when starting the Measure file handling program (MEASFH on SYS*nn* subvolume). The high-order portion of the NEWPROCESS error is included in the message.

Effect. Certain status items may not be measured.

Recovery. See the *System Procedure Errors and Messages Manual* for an explanation of Measure and file-system errors.

0249

```
File error returned from NEWPROCESS while starting the PUP:
```

Cause. The Measure sample required that PUP be started. The NEWPROCESS call returned an error. This is commonly due to a security error on the PUP program file (PUP on SYS*nn* subvolume). The file-system error number is included in the message.

Effect. Certain status items may not be measured.

Recovery. See the *System Procedure Errors and Messages Manual* for an explanation of Measure, NEWPROCESS, and file-system errors.

0250

```
NEWPROCESS error occurred while starting the PUP:
```

Cause. The measurement required PUP be started, but an error was returned from NEWPROCESS while starting PUP. The high-order portion of the NEWPROCESS error is included in the message.

Effect. Certain status items may not be measured.

Recovery. See the *System Procedure Errors and Messages Manual* for an explanation of NEWPROCESS errors.

0251

```
The status configuration file version is not recognized.
```

Cause. This configuration file version is not recognized by the software.

Effect. The configuration file cannot be used.

Recovery. Try creating a new configuration file.

0252

```
SCP program must be owned by SUPER.SUPER and given PROGID
authority.
```

Cause. The requested sample requires that SCP be used. $ZNET is not running, so the server must start SCP. The server is not SUPER.SUPER, so the SCP program file must be owned by SUPER.SUPER and given PROGID authority. $SYSTEM.SYS*nn*.SCP does not have these attributes.

Effect. Certain status items may not be measured.

Recovery. Contact your system administrator to request that the security access of the SCP program file be changed.

0253

```
An IO error occurred on request to the server or PATHMON:
```

Cause. The SEND request resulted in an IO error. The error is indicated by the second number in the message. If an external PATHMON was specified, the IO error may be in the request to the PATHMON. Otherwise the error may be in the request to the server itself.

Effect. ViewPoint may not report correct information.

Recovery. See the *Pathway SCREEN COBOL Reference Manual* for an explanation of SEND errors.

0254

```
Enter "Y" or "N" in the indicated boolean field.
```

Cause. You have specified an incorrect value in the field that is highlighted on your screen.

Effect. No value is selected.

Recovery. Enter a Y or N in the indicated field.

0255

```
The CPU required for this sample is down or does not exist.
```

Cause. The CPU required for this sample is down or does not exist according to PROCESSORSTATUS status values obtained during the sample.

Effect. The CPU item is not measured.

Recovery. Enter an existing CPU on the Status Configuration screen.

0256

```
Enter correct decimal value of MAXIMUM.
```

Cause. The value entered in the “Use Maximum for 100%” field is not correct.

Effect. No value is selected.

Recovery. Enter a correct value.

0257

```
Status server requires a more recent Guardian version.
```

Cause. The version of the operating system is incompatible with the version of the status server.

Effect. No status items are measured.

Recovery. Use a more recent release of the operating system.

0300

```
Please select events and press F6 to acknowledge.
```

Cause. You did not select events that you want to acknowledge.

Effect. No events are acknowledged.

Recovery. Select your events and press F6 to acknowledge them.

0301 (A)

```
Event acknowledged.
```

Cause. The event you selected is acknowledged.

Effect. Information only

Recovery. N/A

0302 (A)

```
Events acknowledged.
```

Cause. The events you selected are acknowledged.

Effect. Information only

Recovery. N/A

0303 (A)

```
Event configuration deleted.
```

Cause. The settings in the current configuration screen are deleted.

Effect. Information only

Recovery. N/A

0304

```
The selected event does not have an appropriate subject.
```

Cause. The first (or other) subject in the event cannot be externalized or does not exist.

Effect. No event is displayed.

Recovery. Select an event that has a known subject. For more information on displaying events with a particular subject, see the description of the Last Events screen in Section 3.

0305

```
Error returned from EMSTEXT:
```

Cause. This error was returned from EMSTEXT. The top half of the EMSTEXT error code was zero.

Effect. The event message may be garbled.

Recovery. See the *EMS Manual* for an explanation of EMSTEXT errors.

0306

```
Please select events and press F11 for detail information.
```

Cause. F11 was pressed when no event was selected.

Effect. No detailed information is displayed.

Recovery. Select an event and press F11 to get detailed information on the event.

0307

```
ZVPT-EVNT-DISP send error:
```

Cause. A SEND to the event collection server resulted in an error.

Effect. The event screen may not be updated correctly.

Recovery. See the *Pathway SCREEN COBOL Reference Manual* for an explanation of SEND errors.

0308

```
ZVPT-EVNT-DISP request error:
```

Cause. An internal error was detected in the event display server.

Effect. The event screen may not be updated correctly.

Recovery. Supply the error numbers when reporting this internal error to a Compaq representative.

0309

```
ZVPT-EVNT-COLL send error:
```

Cause. A SEND to the event collection server resulted in an error.

Effect. The event screen may not be updated correctly.

Recovery. For an explanation of SEND errors, see *Pathway SCREEN COBOL Reference Manual*.

0310

```
ZVPT-EVNT-COLL request error:
```

Cause. The event collection server detected an internal error.

Effect. The event screen may not be updated correctly.

Recovery. Supply the error numbers when reporting this internal error to a Compaq representative.

0311

```
Please press F12 to configure this display.
```

Cause. The Alternate-Events screen is not configured.

Effect. Events are not collected.

Recovery. Press F12 to configure the screen.

0312

```
The selected event is no longer available.
```

Cause. The selected event is no longer in the event cache.

Effect. The event is not available for your particular operation.

Recovery. Select another event.

0313

```
ZVPT-EVNT-COLL error: open table full.
```

Cause. The table where the server keeps track of alternate screens is full.

Effect. No more event displays may be added.

Recovery. Reduce the number of configured alternate event screens.

0314 (A)

```
This page contains all new events.
```

Cause. Old events on the screen have been replaced with all new events.

Effect. Information only.

Recovery. N/A

0315

```
Error returned from EMS distributor:
```

Cause. The EMS distributor returned an error.

Effect. The events screen might not be updated correctly. Or, default source code is not included in your custom filter, and you have supplied default parameter values.

Recovery. See the *EMS Manual* for an explanation of distributor errors. For default source code, see Figure 6-2 in Section 6, "Customizing ViewPoint."

0316

```
Error occurred when opening the event terminal file:
```

Cause. This file-system error occurred when ViewPoint opened the event terminal file.

Effect. The Events Screen might not be updated correctly.

Recovery. See the *System Procedure Errors and Messages Manual* for an explanation of file-system errors.

0317

```
Error occurred when opening the event cache file:
```

Cause. This file-system error occurred when ViewPoint opened the event cache file.

Effect. The Events Screen may not be updated correctly.

Recovery. See the *System Procedure Errors and Messages Manual* for an explanation of file-system errors.

0318

```
Error occurred when opening the action tags file:
```

Cause. This file-system error occurred when opening the action tags file.

Effect. Action events may not be updated correctly.

Recovery. See the *System Procedure Errors and Messages Manual* for an explanation of file-system errors.

0319

```
Error occurred when opening the last event subject file:
```

Cause. This file-system error occurred when opening the last event subject file.

Effect. The Last Events screen may not be updated correctly.

Recovery. See the *System Procedure Errors and Messages Manual* for an explanation of file-system errors.

0320

```
ZVPT-EVNT-DETL Send Error:
```

Cause. A SEND to the event detail server resulted in an error.

Effect. Event detail information may not be available.

Recovery. See the *Pathway SCREEN COBOL Reference Manual* for an explanation of SEND errors.

0321

```
ZVPT-EVNT-DETL Request Error:
```

Cause. The event detail server detected an internal error.

Effect. Event detail information may not be available.

Recovery. Supply the error numbers when reporting this internal error to a Compaq representative.

0322

```
File error occurred on the custom cause/action file:
```

Cause. Customer-provided probable cause and recovery action file could not be read because of a file-system error. The file-system error number is provided.

Effect. Custom event detail information may not be available.

Recovery. See the *System Procedure Errors and Messages Manual* for an explanation of file-system errors.

0323

```
File error occurred on the standard cause/action file:
```

Cause. The probable cause and recommended action file provided by Compaq could not be read because of a file-system error. The file-system error number is provided.

Effect. Event detail information may not be available.

Recovery. See the *System Procedure Errors and Messages Manual* for an explanation of file-system errors.

0324 (A)P

```
No old events were found; continuing with current events.
```

Cause. You tried to display old events when they were not available.

Effect. No events are displayed.

Recovery. Specify more recent events on the event configuration screen.

0325

```
Error occurred when opening the last event cache file:
```

Cause. This file-system error occurred during an OPEN of the last event cache file.

Effect. The Last Events screen might not be updated correctly.

Recovery. See the *System Procedure Errors and Messages Manual* for an explanation of file-system errors.

0326

```
This display is not managed by this event collection server.
```

Cause. The event collection server has been configured not to manage events for the current display. Either the server's Pathway configuration is in error, or the requester is sending its request to the wrong collection server.

Effect. Events are not displayed.

Recovery. Verify that any "ignore" options are set correctly. See Appendix A, "Server Assigns and Params," in this manual for details.

0328

```
The event configuration file version is unrecognized.
```

Cause. The version of this event configuration file is no longer recognized. You cannot use it.

Effect. The current configuration file cannot be used.

Recovery. Try creating a new configuration file.

0329

```
ZVPT-EVNT-CNFG send error:
```

Cause. A SEND to the event configuration server resulted in an error.

Effect. The event configuration file may not be updated.

Recovery. See the *Pathway SCREEN COBOL Reference Manual* for an explanation of SEND errors.

0330

```
ZVPT-EVNT-CNFG request error:
```

Cause. The event configuration server detected an internal error.

Effect. The event configuration file may not be updated.

Recovery. Supply the error numbers when reporting this internal error to a Compaq representative.

0331

```
An error occurred during a read of the event configuration
file:
```

Cause. ViewPoint reported a file-system error while updating the status configuration file. The error reported is the second number in the message.

Effect. The current configuration file may not be used.

Recovery. See the *System Procedure Errors and Messages Manual* for an explanation of file-system errors.

0332

```
Please enter the name of an event configuration file.
```

Cause. An invalid file name was entered as the event configuration file.

Effect. No event configuration file is created.

Recovery. Specify a valid event configuration file name.

0333

```
An error occurred during an update to the event configuration
file:
```

Cause. A file-system error was reported while ViewPoint was updating the status configuration file. The error reported is shown as the second number in the message.

Effect. The values entered in the event configuration file may not be saved.

Recovery. See the *System Procedure Errors and Messages Manual* for an explanation of file-system errors.

0334

```
Please enter a correct event configuration filename.
```

Cause. The file name you entered is not valid.

Effect. No name is selected.

Recovery. Enter a valid file name.

0335

```
Error occurred when creating the event configuration file:
```

Cause. A file-system error was reported while ViewPoint was creating the status configuration file. The error reported is shown as the second number in the message.

Effect. The values entered in the event configuration file may not be saved.

Recovery. See the *System Procedure Errors and Messages Manual* for an explanation of file-system errors.

0336

```
Please enter a correct filter file name:
```

Cause. The name entered as the filter file is invalid.

Effect. Events are not displayed.

Recovery. Enter a valid filter file name.

0337

```
No events are available after the date provided:
```

Cause. The After field on the event configuration screen was set to a date following the last events in the event cache.

Effect. No events can be displayed on the Alternate Events screen.

Recovery. Set the After field to an earlier date.

0338

```
Please correct the starting date; no events are available.
```

Cause. The After field on the event configuration screen was set to a date following the last events in the event cache.

Effect. No events can be displayed on the Alternate Events screen.

Recovery. Set the After field to an earlier date.

0339

```
Please enter a positive or zero delay time.
```

Cause. An invalid delay time was entered on the configuration screen.

Effect. No events are displayed.

Recovery. Enter a delay time that is either positive or zero.

0340

```
Please use the correct number of pages desired.
```

Cause. An invalid page number was entered.

Effect. No page number is selected.

Recovery. Enter a number between 1 and 750.

0341 (A)

```
Event configuration updated.
```

Cause. The current event configuration has been saved.

Effect. Information only

Recovery. N/A

0342

```
Please use a correct filter file name.
```

Cause. You entered an invalid filter file name.

Effect. No events are collected.

Recovery. Specify a valid filter file name.

0343

```
Please use a correct collector or log file name.
```

Cause. You entered an invalid collector or log file name.

Effect. No events are displayed.

Recovery. Specify a valid collector or log file name.

0344 (A)

```
Press F14 again to create the new event-configuration file.
```

Cause. This confirmation message is displayed after a new file name is entered and F14 is pressed (Profile screen).

Effect. Information only

Recovery. N/A

0345 (A)

```
The event configuration file has been changed.
```

Cause. The event configuration file has been changed. This is reported from the Profile screen when the configuration file field is modified.

Effect. Information only

Recovery. N/A

0346 (A)

```
The event configuration file has been created.
```

Cause. The event configuration file has been created. This is reported from the Profile screen when the configuration file field is modified and the confirmation flag is set.

Effect. Information only

Recovery. N/A

0347

```
Please press F12 to configure the event display.
```

Cause. This event display has not been configured.

Effect. Events are not displayed.

Recovery. Configure the event display by pressing F12 to go to the Event Configuration screen.

0348 (A)

```
The event configuration has been changed.
```

Cause. The event configuration for this display has been changed as a result of filling in the Event Configuration Screen and pressing F12.

Effect. Information only

Recovery. N/A

0349 (A)

```
Alternate event acknowledged.
```

Cause. You acknowledged an event by pressing F6.

Effect. Information only

Recovery. N/A

0350 (A)

```
Alternate events acknowledged.
```

Cause. You acknowledged an event by pressing F6.

Effect. Information only

Recovery. N/A

0351

```
Error occurred while reading the event terminal file:
```

Cause. This file-system error occurred while ViewPoint was reading the event terminal file.

Effect. Events may not be displayed.

Recovery. See the *System Procedure Errors and Messages Manual* for an explanation of file-system errors.

0352

```
Error occurred while writing to the event terminal file:
```

Cause. This file-system error occurred while ViewPoint was writing to the event terminal file.

Effect. Events may not be displayed.

Recovery. See the *System Procedure Errors and Messages Manual* for an explanation of file-system errors.

0353

```
Error occurred while reading the event cache file:
```

Cause. This is an internal error.

Effect. Events may not be displayed.

Recovery. Report this internal error to a Compaq representative, and be sure to supply the error numbers.

0354

```
Error occurred while writing to the event cache file:
```

Cause. This is an internal error.

Effect. Events may not be displayed.

Recovery. Report this internal error to a Compaq representative, and be sure to supply the error numbers.

0355

```
Error occurred while reading the event action tags file:
```

Cause. This file-system error occurred while ViewPoint was reading the event action tags file.

Effect. Events may not be accurately displayed on your screen.

Recovery. See the *System Procedure Errors and Messages Manual* for an explanation of file-system errors.

0356

```
Error occurred while writing to the event action tags file:
```

Cause. This file-system error occurred while ViewPoint was writing to the event action tags file.

Effect. Events may not be accurately displayed on your screen.

Recovery. See the *System Procedure Errors and Messages Manual* for an explanation of file-system errors.

0357

```
Error occurred while reading the last events cache file:
```

Cause. This file-system error occurred while ViewPoint was reading the last-events cache file.

Effect. Last events may not be accurately displayed on your screen.

Recovery. See the *System Procedure Errors and Messages Manual* for an explanation of file-system errors.

0358

```
Error occurred while writing to the last events cache file:
```

Cause. This file-system error occurred while ViewPoint was writing to the last events cache file.

Effect. Last events may not be accurately displayed on your screen.

Recovery. See the *System Procedure Errors and Messages Manual* for an explanation of file-system errors.

0359

```
Error occurred while reading the last events subject file:
```

Cause. This file-system error occurred while ViewPoint was reading the last events subject file.

Effect. Last events may not be accurately displayed on your screen.

Recovery. See the *System Procedure Errors and Messages Manual* for an explanation of file-system errors.

0360

```
Error occurred while writing to the last events subject file:
```

Cause. This file-system error occurred while ViewPoint was writing to the last events subject file.

Effect. Last events may not be accurately displayed on your screen.

Recovery. See the *System Procedure Errors and Messages Manual* for an explanation of file-system errors.

0361

```
An event contains an incorrect timestamp.
```

Cause. An event contains an incorrect timestamp in the header. The CONVERTTIMESTAMP error is returned.

Effect. The event is not displayed.

Recovery. None

0362

```
Error occurred while starting an event distributor:
```

Cause. A NEWPROCESS or CREATEPROCESSNAME error occurred while attempting to start an event distributor.

Effect. Events may not be displayed.

Recovery. If this is a NEWPROCESS error, the error is returned. See the *System Procedure Errors and Messages Manual* for an explanation of NEWPROCESS errors.

0363

```
Error occurred while opening an event distributor:
```

Cause. A file-system error occurred when ViewPoint attempted to open a started event distributor.

Effect. Events may not be displayed.

Recovery. See the *System Procedure Errors and Messages Manual* for an explanation of file-system errors.

0364

```
Error occurred while writing to an event distributor:
```

Cause. A file-system error occurred when ViewPoint attempted to write to an event distributor.

Effect. Events might not be displayed.

Recovery. See the *System Procedure Errors and Messages Manual* for an explanation of file-system errors.

0365

```
Error occurred while reading from an event distributor:
```

Cause. A file-system error occurred when attempting to read from an event distributor.

Effect. Events may not be displayed.

Recovery. See the *System Procedure Errors and Messages Manual* for an explanation of file-system errors.

0366

```
The ZVPT-EVNT-COLL server is out of distributor pool space.
```

Cause. The ZVPT-EVNT-COLL server ran out of internal I/O pool space for communication with event distributors.

Effect. No more event distributors can be started.

Recovery. Decrease the number of event distributors running.

0367

```
The ZVPT-EVNT-COLL server has a corrupt distributor pool
space.
```

Cause. The ZVPT-EVNT-COLL server could not put an I/O buffer back into the pool because the pool is now corrupt.

Effect. Events might not be displayed.

Recovery. The server must be restarted.

0368

```
Error occurred while reading the terminal list file:
```

Cause. A file-system error occurred when reading the terminal list file. The error reported is shown as the second number in the message.

Effect. Events may not be updated.

Recovery. See the *System Procedure Errors and Messages Manual* for an explanation of file-system errors.

0369

```
Error occurred while writing to the terminal list file:
```

Cause. A file-system error occurred when writing to the terminal list file. The error reported is shown as the second number in the message.

Effect. Events may not be updated.

Recovery. See the *System Procedure Errors and Messages Manual* for an explanation of file-system errors.

0370

```
The TCP name associated with this terminal is incorrect.
```

Cause. A TCP name cannot be found for this terminal. This is an internal error.

Effect. Screens may not be updated.

Recovery. Report this internal error to a Compaq representative, and be sure to supply the error numbers.

0371

```
Error occurred while opening the TCP for this terminal:
```

Cause. A file-system error occurred while opening TCP for this terminal. The error reported is shown as the second number in the message.

Effect. ViewPoint screens may not be available.

Recovery. See the *System Procedure Errors and Messages Manual* for an explanation of file-system errors.

0372

```
Error occurred while writing to the TCP for this terminal:
```

Cause. A file-system error occurred while writing to the TCP for this terminal. The error reported is shown as the second number in the message.

Effect. ViewPoint screens may not be available.

Recovery. See the *System Procedure Errors and Messages Manual* for an explanation of file-system errors.

0373

```
Error occurred while reading the TCP for this terminal:
```

Cause. A file-system error occurred while reading the TCP for this terminal. The error reported is shown as the second number in the message.

Effect. ViewPoint screens may not be available.

Recovery. See the *System Procedure Errors and Messages Manual* for an explanation of file-system errors.

0374

```
An incorrect unsolicited message reply length was received:
```

Cause. An invalid unsolicited message reply length was received.

Effect. Events may not be displayed.

Recovery. See the *Pathway Application Programming Guide* for an explanation of the error.

0375

```
Error occurred when opening the marked event file:
```

Cause. A file-system error occurred while opening the marked event file. The error reported is shown as the second number in the message.

Effect. Events may not be marked correctly.

Recovery. See the *System Procedure Errors and Messages Manual* for an explanation of file-system errors.

0376

```
Error occurred while reading the marked event file:
```

Cause. A file-system error occurred while reading the marked event file. The error reported is shown as the second number in the message.

Effect. Events may not be marked correctly.

Recovery. See the *System Procedure Errors and Messages Manual* for an explanation of file-system errors.

0377

```
Error occurred while writing the marked event file:
```

Cause. A file-system error occurred while writing the marked event file. The error reported is shown as the second number in the message.

Effect. Events may not be marked correctly.

Recovery. See the *System Procedure Errors and Messages Manual* for an explanation of file-system errors.

0378

```
Can't delete the event.
```

Cause. The event selected for deletion could not be deleted.

Effect. The event is not deleted.

Recovery. Select a different event or change the flag settings in the EXTRAS-DELETE program.

0379

```
Can't delete some of the events.
```

Cause. The events selected for deletion could not be deleted.

Effect. The events are not be deleted.

Recovery. Select different events or change the flag settings in the EXTRAS-DELETE program.

0380

```
Can't delete the events.
```

Cause. The events selected for deletion could not be deleted.

Effect. No event is deleted.

Recovery. Select different events or change the flag settings in the EXTRAS-DELETE program.

0381

```
The events are different from those on the most recently
displayed page.
```

Cause. The events on this page of the event display are different from those on the most recently displayed page of the event screen, although the page number is the same.

Effect. None

Recovery. Specify a different log file or a valid date in the current log file.

0382

```
Please use a correct system name.
```

Cause. The system name you entered is invalid.

Effect. No event is displayed.

Recovery. Enter a valid system name.

0383

```
Please use a correct subsystem name.
```

Cause. The system name you entered is invalid.

Effect. No event is displayed.

Recovery. Enter a valid system name.

0384

```
Please use a correct event number.  The event number should
be an integer.
```

Cause. The event number you entered is invalid.

Effect. No event is displayed.

Recovery. Enter a valid event number.

0385

```
Please use a correct process name.
```

Cause. The process name you entered is invalid. The process name should be either symbolic (for example, $VP) or in the form “CPU, PIN.”

Effect. No event is displayed.

Recovery. Enter a valid process name or process ID.

0386

```
Please use a correct event text.
```

Cause. The event text is not properly enclosed in quotation marks.

Effect. No event is displayed.

Recovery. Enter event text that is properly enclosed in quotation marks. See the description of the Event Configuration screen in Section 3, Definition of ViewPoint Screens for information on enclosing event text in quotation marks.

0387

```
Please use either selected or discarded.
```

Cause. The Selected and Discarded options were both selected on the Event Configuration screen. Only one option can be selected at a time.

Effect. No event is displayed.

Recovery. Specify either Selected or Discarded.

0388

```
Please use a correct value for token.
```

Cause. The value of token is incorrect for the selected type (file, string, or number).

Effect. No event is displayed.

Recovery. Enter a value that is correct for the selected type.

0389

```
A maximum of 5 collector process names may be specified.
```

Cause. A log file was specified along with one or more other log file or collector process names. Only one log file or as many as five collector process names can be specified.

Effect. No events are displayed.

Recovery. Enter as many as five collector process names or one log file name.

0390

```
One log file name or 5 collector process names may be
specified.
```

Cause. A log file was specified along with one or more other log file or collector process names. Only one log file or as many as five collector process names can be specified.

Effect. No events are displayed.

Recovery. Enter as many as five collector process names or one log file name.

0391

```
The specified page number is higher than the last page number
```

Cause. The specified page number is higher than the last page of the event display; therefore, the last page was displayed.

Effect. The last page is displayed.

Recovery. Specify a page number that is not higher than the last page.

0392

```
Please select only one type for the custom token
```

Cause. More than one type was selected for a custom token. Only one of the types (file, string, or number) can be specified.

Effect. No events are displayed.

Recovery. Enter one type for the token.

0393

```
Please specify a type for the custom token.
```

Cause. A value was specified for the custom token but its type (file, string, or number) was not selected.

Effect. No events are displayed.

Recovery. Select a type for the token.

0394

```
Please specify a value for the custom token.
```

Cause. You selected a type (file, string, or number) for a custom token, but did not specify a value for it.

Effect. No events are displayed.

Recovery. Enter a value appropriate for the type selected.

0395

```
Please specify a correct file name for the custom token.
```

Cause. An invalid file name was specified as the value for the custom token.

Effect. No events are displayed.

Recovery. Enter a valid file name for the custom token.

0396

```
Please specify a correctly quoted string value for the custom
token.
```

Cause. A string value was specified for the custom token, but the value is not properly enclosed in quotes.

Effect. No events are displayed.

Recovery. Enter a string value that is properly enclosed in quotes.

0397

```
Please specify a correct number as the value for the custom
token.
```

Cause. You specified an illegal number value for the custom token. The value is either incorrectly specified or is not in the range -2,147,483,648 through +2,147,483,647.

Effect. No events are displayed.

Recovery. Specify a valid value.

0398

```
A maximum of 1 log file names may be specified.
```

Cause. More than one log file name was specified; only one log file is allowed.

Effect. No events are displayed.

Recovery. Enter only one log file.

0399

```
Please use a correct collector or log file name.
```

Cause. The name specified for a collector or log file is not valid.

Effect. No events are displayed.

Recovery. Specify a valid collector or log file name.

0400

```
Please enter a known event subject and press F4.
```

Cause. The subject name entered on the option line does not match any of the event-message subjects.

Effect. No events are displayed.

Recovery. Enter a known event subject. For more information on displaying events with a particular subject, see the description of the Last Events screen in Section 3, “Definition of ViewPoint Screens.”

0401

```
The first collector could not be accessed and was skipped.
```

Cause. The first collector specified is inaccessible and cannot be used.

Effect. No events from the first collector are displayed.

Recovery. Specify a collector that is accessible.

0402

```
The second collector could not be accessed and was skipped.
```

Cause. The second collector specified is inaccessible and cannot be used.

Effect. No events from the second collector are displayed.

Recovery. Specify a collector that is accessible.

0403

```
The third collector could not be accessed and was skipped.
```

Cause. The third collector specified is inaccessible and cannot be used.

Effect. No events from the third collector are displayed.

Recovery. Specify a collector that is accessible.

0404

```
The fourth collector could not be accessed and was skipped.
```

Cause. The fourth collector specified is inaccessible and cannot be used.

Effect. No events from the fourth collector are displayed.

Recovery. Specify a collector that is accessible.

0405

```
The fifth collector could not be accessed and was skipped.
```

Cause. The fifth collector specified is inaccessible and cannot be used.

Effect. No events from the fifth collector are displayed.

Recovery. Specify a collector that is accessible.

0406

```
A duplicate collector name was specified.
```

Cause. A collector name was specified that is the same name used for an existing collector.

Effect. No events are displayed.

Recovery. Specify a different collector name.

0407

```
A security error occurred while reading the EMS Log File.
```

Cause. You do not have security access to the EMS log file.

Effect. No events are displayed.

Recovery. Contact your system administrator to request that the security access of the EMS log file be changed.

0408

```
A error occurred while reading the EMS Log File:
```

Cause. A file-system error occurred while reading the EMS log file.

Effect. No events are displayed.

Recovery. See the *System Procedure Errors and Messages Manual* for an explanation of file-system errors.

0409

```
A error occurred while starting an event distributor:
```

Cause. A file error occurred during NEWPROCESS of an event distributor. The error returned is a file error pertaining to the program file, swap file, or process name.

Effect. No events are displayed.

Recovery. See the *System Procedure Errors and Messages Manual* for an explanation of file-system errors.

0410

```
An event view display option is required.  "None" is assumed.
```

Cause. No option was selected from a row of options under the Event View of the Event Configuration Screen. “None” is assumed as a default for this row.

Effect. Certain events are not displayed.

Recovery. Select “All,” “Outstanding,” or “Acknowledged” for the view of the event that you desire.

0411

```
Select an option other than "None" to view events.
```

Cause. No option was selected or “None” was selected for all three rows of options under the Event View of the Event Configuration screen.

Effect. No events are displayed.

Recovery. Select at least one option other than “None.”

0412

```
Select a single display option for each event type.
```

Cause. More than one display option was selected on a single row of options under the Event View of the Event Configuration screen. Only one option may be selected on each row.

Effect. No events are displayed.

Recovery. Select only one display option on each row.

0413

```
The ZVPT-EVNT-COLL server is unable to allocated memory for
cache.
```

Cause. The ZPVT-EVNT-COLL server encountered an error while trying to allocate or change the size of the memory segment for the event cache. The error is reported in a ViewPoint event which is logged to $0.

Effect. Events may not be displayed.

Recovery. Restart the ZVPT-EVNT-COLL server.

0414

```
The ZVPT-EVNT-COLL server memory event cache has been
corrupted.
```

Cause. The ZPVT-EVNT-COLL server encountered an error while managing the memory event cache. A ViewPoint event is logged to $0 which describes this error.

Effect. Events may not be displayed.

Recovery. Restart the ZVPT-EVNT-COLL server.

0415

```
No Event Information For DSMS Object
```

Cause. No event information was found on the Last Events for the object name passed from DSMS. This can only occur in the Integrated Operations Console (IOC) environment.

Effect. Information only

Recovery. N/A

0502

```
An internal error was returned from SSGET on the event.
```

Cause. An SSGET on the event buffer resulted in an error. The error number is included in the message.

Effect. Events may not be displayed.

Recovery. See the *SPI Programming Manual* for an explanation of SPI procedure errors.

C ViewPoint Event Messages

ViewPoint Event Logging

ViewPoint reports errors detected during the display of screens on a terminal on the advice line of the terminal. However, ViewPoint reports errors detected by servers that run in an asynchronous mode to terminal operation as event messages through the Event Management Service (EMS). An example of such a server is the garbage collection server (ZVPT-GARB-COLL); this server runs at a scheduled time and performs database maintenance.

ViewPoint builds the event messages in an SPI message in a format specified by EMS and sends the message to event collector $0 on the system where the ViewPoint PATHMON process is running. The event is sent using the file-system WRITEREAD procedure.

The events that ViewPoint can report are described on the following pages. The values in parentheses are the event numbers.

ZVPT-EVT-IO-ERROR (1)

The garbage collection server or the event collection server generates this event when either server encounters an irrecoverable I/O error.

Tokens

Token Name	Value Assigned
ZVPT-TKN-COMPONENT	ZVPT-VAL-GARBAGE-COLLECT or ZVPT-VAL-EVENT-COLLECT
ZVPT-TKN-SEVERITY	ZVPT-VAL-SEVERITY-ERROR
ZVPT-TKN-PATHMONNAME	*process file name*
ZVPT-TKN-FNAME	*file name*
ZEMS-TKN-EMPHASIS	TRUE
ZEMS-TKN-GENTIME	(EMS provides)
ZEMS-TKN-CRTPID	(EMS provides)
ZEMS-TKN-CONSOLE-PRINT	TRUE
ZEMS-TKN-EVENTNUMBER	ZVPT-EVT-IO-ERROR
ZFIL.ZSPI-TKN-ERRLIST	
ZSPI-TKN-ERROR	*error number*
ZSPI-TKN-PROC-ERR	*file system procedure*
ZFIL-TKN-OBJECTFILE	*server object file name*
ZFIL-TKN-FILENAME	*name of file*

Subject

ZVPT-TKN-PATHMONNAME	*process file name*

Event Text Example

```
VIEWPT \NCN.$ZVPT: ERROR Garbage-Collect File I/O Error
on \NCN.$DATA.VPTDATA.EVNTTERM, Error 60
```

Cause

One of the ViewPoint data files has been removed, damaged, or made inaccessible. The file-management error number and file name in the message indicate the specific problem.

Effect

If the garbage collection server encounters this error, the cleanup operation is stopped. If an event collection server encounters this error, event displays are no longer available and users might be unable to enter ViewPoint.

Recovery

The error number in the message determines the needed repair for the file. The data files can be recreated using FUP commands in the file FUPCMDS on the installed data subvolume. Recreating a file destroys the current contents.

ZVPT-EVT-OPEN-ERROR (2)

The garbage collection and event collection servers generate this event when either server is unable to open a data file.

Tokens

Token Name	Value Assigned
ZVPT-TKN-COMPONENT	ZVPT-VAL-GARBAGE-COLLECT or ZVPT-VAL-EVENT-COLLECT
ZVPT-TKN-SEVERITY	ZVPT-VAL-SEVERITY-ERROR
ZVPT-TKN-PATHMONNAME	*process file name*
ZVPT-TKN-FNAME	*file name*
ZEMS-TKN-EMPHASIS	TRUE
ZEMS-TKN-GENTIME	(EMS provides)
ZEMS-TKN-CRTPID	(EMS provides)
ZEMS-TKN-CONSOLE-PRINT	TRUE
ZEMS-TKN-EVENTNUMBER	ZVPT-EVT-OPEN-ERROR
ZFIL.ZSPI-TKN-ERRLIST	
ZSPI-TKN-ERROR	*error number*
ZSPI-TKN-PROC-ERR	ZFIL-CMD-OPEN
ZFIL-TKN-OBJECTFILE	*server object filename*
ZFIL-TKN-FILENAME	*name of file*

Subject

ZVPT-TKN-PATHMONNAME	*process file name*

Event Text Example

```
VIEWPT \NCN.$ZVPT: ERROR Garbage-Collect File Open Error
on \NCN.$DATA.VPTDATA.EVNTCACH, Error 11
```

Cause

One of the ViewPoint data files has been removed, damaged, or made inaccessible. The file-management error number and file name in the message indicate the specific problem.

Effect

If the garbage collection server encounters this error, ViewPoint stops the cleanup operation. If an event collection server encounters the error, event displays are not available and users might be unable to enter ViewPoint.

Recovery

The error number in the message determines the needed repair for the file. If the file is missing (error number 11), then the Pathway configuration assigns for the server might be directing it to the wrong file name. The PATHCOM INFO command can be used to check the configuration. The data files can be recreated using FUP commands in the file FUPCMDS on the installation subvolume. Recreating the files destroys the current contents.

ZVPT-EVT-SSGET-ERROR (4)

The garbage collection server or the event collection server generates this event when either server receives an error while accessing tokens within event messages in an event cache.

Tokens

Token Name	Value Assigned
ZVPT-TKN-COMPONENT	ZVPT-VAL-GARBAGE-COLLECT or ZVPT-VAL-EVENT-COLLECT
ZVPT-TKN-SEVERITY	ZVPT-VAL-SEVERITY-ERROR
ZVPT-TKN-PATHMONNAME	*process file name*
ZEMS-TKN-EMPHASIS	TRUE
ZEMS-TKN-GENTIME	(EMS provides)
ZEMS-TKN-CRTPID	(EMS provides)
ZEMS-TKN-CONSOLE-PRINT	TRUE
ZEMS-TKN-EVENTNUMBER	ZVPT-EVT-SSGET-ERROR
ZVPT-TKN-SPIERROR	*SSGET error code*

Subject

ZVPT-TKN-PATHMONNAME	*process file name*

Event Text Example

```
VIEWPT \NCN.$ZVPT: ERROR Garbage-Collect SSGET returned
error -8
```

Cause

The ViewPoint server got an error while getting a core token from an event in an event cache data file. This is an internal error indicating improperly formatted events.

Effect

If the garbage collection server encounters this error, ViewPoint stops the cleanup operation. If an event collection server encounters the error, event displays are no longer available and users might not be able to enter ViewPoint.

Recovery

This problem might be due to incompatible versions of installed software. This problem should be reported to support personnel.

ZVPT-EVT-TOO-LONG (5)

The event collection server generates this event when it receives an event message that is too large to be recorded in an event cache (greater than 3900 bytes). This event replaces the too-large event in the ViewPoint event cache and appears on the ViewPoint event displays. This event is not sent to the EMS collector and does not appear on event logs. The original too-large event appears on the event log.

Tokens

Token Name	Value Assigned
ZVPT-TKN-COMPONENT	ZVPT-VAL-EVENT-COLLECT
ZVPT-TKN-SEVERITY	ZVPT-VAL-SEVERITY-ERROR
ZVPT-TKN-PATHMONNAME	*process file name*
ZEMS-TKN-EMPHASIS	TRUE
ZEMS-TKN-GENTIME	(EMS provides)
ZEMS-TKN-CRTPID	(EMS provides)
ZEMS-TKN-CONSOLE-PRINT	TRUE
ZEMS-TKN-EVENTNUMBER	ZVPT-EVT-TOO-LONG
ZVPT-TKN-TEXT	*text of long event*

Subject

ZVPT-TKN-PATHMONNAME	*process file name*

Event Text Example

```
89-03-30 10:41 \NCN.$XYZ TANDEM.SUBSYS.C20 123 formatted
text for long event ...
```

Cause

The event collection server received an event message that is too long to be recorded in an event cache (greater than 3900 bytes). The subsystem and event number of the large event are indicated by the event text.

Effect

This event replaces the too-large event in the ViewPoint event cache. It does not appear on the ViewPoint event displays; it is sent to the EMS collector. Also, it does not appear in event logs. The original too-large event appears in the event log.

Recovery

A very large event message could indicate erroneous event creation by the subsystem or application that generated the event. If so, the event message should be corrected in the subsystem or application so that ViewPoint can correctly record the event.

ZVPT-EVT-EDITREAD-ERROR (6)

The event collection server generates this event when it receives an error while reading edit files that supply collector configuration information.

Tokens

Token Name	Value Assigned
ZVPT-TKN-COMPONENT	ZVPT-VAL-EVENT-COLLECT
ZVPT-TKN-SEVERITY	ZVPT-VAL-SEVERITY-ERROR
ZVPT-TKN-PATHMONNAME	*process file name*
ZEMS-TKN-EMPHASIS	TRUE
ZEMS-TKN-GENTIME	(EMS provides)
ZEMS-TKN-CRTPID	(EMS provides)
ZEMS-TKN-CONSOLE-PRINT	TRUE
ZEMS-TKN-EVENTNUMBER	ZVPT-EVT-EDITREAD-ERROR
ZFIL.ZSPI-TKN-ERRLIST	
ZSPI-TKN-ERROR	*error number*
ZSPI-TKN-PROC-ERROR	0
ZFIL-TKN-OBJECTFILE	*server object file name*
ZFIL-TKN-FILENAME	*file name*

Subject

ZVPT-TKN-PATHMONNAME	*process file name*

Event Text Example

```
ViewPoint \NCN.$ZVPT: ERROR Event-Collect Editread
or format error on $DATA.VIEWPT.COLLIST, Error 0
```

Effect

The event collection server ends abnormally after generating this error. The default is set for the server to restart as many as five times; if the error persists, users are not able to enter ViewPoint or use the event screens.

Recovery

The problem indicated by the error number should be corrected. For syntax errors, refer to the rules specified in this manual. When the error is corrected, the server can be restarted with a PATHCOM START SERVER command.

ZVPT-EVT-MEM-ERROR (7)

The event collection server generates this event when it encounters an unrecoverable memory allocation error.

Tokens

Token Name	Value Assigned
ZVPT-TKN-COMPONENT	ZVPT-VAL-EVENT-COLLECT
ZVPT-TKN-SEVERITY	ZVPT-VAL-SEVERITY-ERROR
ZVPT-TKN-PATHMONNAME	*process file name*
ZEMS-TKN-EMPHASIS	TRUE
ZEMS-TKN-GENTIME	(EMS provides)
ZEMS-TKN-CRTPID	(EMS provides)
ZEMS-TKN-CONSOLE-PRINT	TRUE
ZEMS-TKN-EVENTNUMBER	ZVPT-EVT-MEM-ERROR
ZFIL.ZSPI-TKN-ERRLIST	
ZSPI-TKN-ERROR	*error number*
ZSPI-TKN-PROC-ERROR	(operating system procedure id)
ZVPT-TKN-SEGMENTID	*memory segment id*
ZFIL-TKN-FILENAME	*file name*

Subject

ZVPT-TKN-PATHMONNAME	*process file name*

Event Text Example

```
ViewPoint \NCN.$ZVPT: ERROR Event-Collect Memory
allocation error, Procedure <proc name>, Error <nn>,
Segment <mm>
```

Probable Cause

The event collection server generates this event when it encounters a memory allocation error in managing the event display. The operating system procedure that produced the

error and the error number received are included in the message. The segment ID indicates which event display produced the error:

```
1024      => primary event display
1 to 1023 => an alternate event display
```

Effect

ALLOCATESEGMENT errors for alternate event displays cause the alternate event display to stop; the alternate event display must be reconfigured. All other errors cause the event collection server to end abnormally after generating this error. The default is for the server to restart as many as five times. If the error persists, users are not able to enter ViewPoint or use the event screens.

Recovery

ALLOCATESEGMENT and RESIZESEGMENT errors, which indicate that segment space could not be allocated, can be corrected by configuring a smaller cache size for the event display, or by correcting the file system error. Other errors should be reported to your Compaq representative.

D Sample Custom Status Server

This appendix contains the complete source code of a custom status server. If you are a programmer who needs to write a custom server for your ViewPoint application, you can use this code as an example; or, if it suits your purpose, you can use this code as is or extend it. The documentation for the sample server includes instructions for adding status items.

The server displays status information on the ViewPoint Network Status Summary screen and also collects statistical information about the status items it displays.

Refer to Section 7, Installation, Configuration, and Startup for an example of the PATHCOM commands needed to add this server (or any new server) to your Pathway configuration file (PATHDEFS).

Custom Server Code

```
?NOCODE, SYMBOLS, INSPECT, SAVEABEND, NOMAP
?PAGE "CUSTOMIZED STATUS COLLECTION SERVER - OVERALL DESCRIPTION"
!*************************************************************************
!* Version #            Description of Modification                Date    *
!*                                                                         *
!*   1.0              FIRST GENERAL AVAILABILITY RELEASE            25DEC87  *
!*   1.1              Modify Scan Next for automatic roll over      03FEB88  *
!*************************************************************************
!* PROGRAM NAME: USERSTAT    T2001C00^03FEB88
!*
!* FUNCTIONAL DESCRIPTION:
!*
!* This server collects status information for the Viewpoint network status
!* screen.  This is written as a context-free dynamic Pathway server.
!* It is not - in itself - continuous.  A request in progress when the
!* server fails will be retried after recreation of the server.
!*
!* Three basic types of requests are made of the status server: the first
!* type (SCAN) retrieves information about items which the server
!* supports, a second type (CHECK) checks the configuration of an item
!* which is about to be added, a third type (SAMPLE) retrieves data on
!* the current state of an item.  The scan and check requests are made
!* from the Status item configuration screen; the sample request is made
!* from the status screen - after the items have been configured.
!*
!* This server retains a list of the status item's which it supports
!* (item^array).  This list is searched when the SCAN-ITEM request is
!* made; it is stepped through by successive SCAN-NEXT-ITEM requests.
!* Each status item is represented by a structure STATUS-ITEM.  This
!* structure is included in several of the request and reply messages.
!* Default values for the fields in the STATUS-ITEM structure are saved
!* in item^array and are provided when the structure is returned in
!* replies to the SCAN-ITEM and SCAN-NEXT-ITEM requests.  These values
!* may be changed from the Status item configuration screen; the changed
!* values are checked by the server in response to the CHECK-ITEM
!* request.
!*
!* The SAMPLE-ITEM request must return status values.  These values
!* are returned in 64-bit binary entities.  There are two basic
!* types of status values:  busy timers and counters.  The value
!* sampled from a busy timer is a count of microseconds that the object
!* was busy.  This value is a snapshot of the busy timer; successive
!* snapshots are used to compute a percentage.  Counters are scaler
!* values which are used directly.  They are a count of objects up,
!* or throughput counts.
!*
!* Counter values may be fractions; a SCALE field is included in the
!* STATUS-ITEM structure.  It's value (0,1, or 2) indicates the number of
```

```
!* decimal positions which are assumed in the count.  SCALE is a static
!* property of an item which can not be changed by configuration.
!*
!* The MAXIMUM-VALUE field is optionally set to contain the maximum value
!* a counter could attain.
!*
!* A timestamp is supplied with each sample.  This timestamp is the
!* four-word, Julian-date-based, microsecond resolution timestamp at the
!* time of the sample (or as soon as possible after the sample).  This
!* timestamp value is returned by the JULIANTIMESTAMP system procedure
!* call.
!**************************************************************************
?PAGE "HOW TO ADD STATUS ITEMS TO THIS STATUS SERVER?"
!**************************************************************************
!*
!*    *****   How to add Status Items to this status server ?   *****
!*
!*  1- Modify the Defined^Items DEFINE in section STATUS COLLECTION SERVER
!*     GLOBAL VARIABLES AND DEFINITIONS ( Around line 231 )
!*
!*     Ex: Currently Defined^Items is = 6 so you should change the 6 for
!*         a 7.
!*
!*  2- Add the definition of your new status item in PROC Server^Initialize
!*     using the Allocate^Item DEFINE. ( Around line 385 ).
!*
!*     Ex: Let say you want to add an object like File Extent Usage.
!*         i.e. The number of extent currently allocated !
!*
!*         Allocate^Item ( 7,              index
!*                         "File Extent Usage",   description
!*                         "FILE-EXTENT-USAGE",   ITEM TYPE
!*                         "<Filename>",   object name
!*                         100F,           value for 100
!*                         "Y",            use maximum ?
!*                         "N",            reverse signify
!*                         0,              scale
!*                         0,              low threshold
!*                         100,            high threshold
!*                         "N", "N",       low/high threshold enabled
!*                         "CV"            type (busy-timer or counter-value)
!*                         );
!*
!*  3- You will have to enter a new CASE option in the procedure
!*     Check^Item. This CASE line will call a procedure to verify
!*     the validity of the new status item being configured. ( Line 982 ).
!*
!*     Ex:  7 -> Return^Error := Check^Valid^File^Name ( Request, Reply );
!*
!*  4- If the procedure does not exist like Check^Valid^File^Name ,
!*     you will have to write one.
!*
!*  5- You will have to enter a new CASE option in the procedure
!*     Sample^Item. This CASE line will call a procedure to calculate
!*     the information that will be displayed back on the Viewpoint
!*     status screen. ( Line 1029 ).
!*
!*     Ex:  7 -> Return^Error := Sample^File^Extent^Usage ( Request, Reply );
!*
!*  6- Code the procedure Sample^File^Extent^Usage.
!*
!*  7- You are now ready to include your server into the Viewpoint
!*     configuration file and see your new status item on the screen !
!*
!**************************************************************************
?PAGE "GENERAL DEFINES AND LITERALS"
------------------------------------------------------------------------------
--
-- We will present here some useful DEFINES and LITERAL that will be used
-- in this program to increase the readability of this code. At least we hope !
--
------------------------------------------------------------------------------
DEFINE  Word      = INT#;
LITERAL True      = -1;
```

```
LITERAL False     =  0;
LITERAL Fname^Wsz = 12;
LITERAL Recv^User^Msg             = 0; ! A user message was read on $RECEIVE.
LITERAL Recv^System^Msg^From^User = 1; ! A system message related to an
                                       ! opener was read from $RECEIVE.
                                       ! Typically an OPEN, CLOSE, SETMODE
                                       ! or CONTROL message.
LITERAL Recv^End^Of^File          = 2; ! The server should terminate.  No
                                       ! message is returned.
DEFINE
  To^Bsz( x )  = ((x) * 2)#,           ! Compute str len from word len
  To^Wsz( x )  = (((x) + 1) / 2)#,     ! Compute word len from str len
  Len^Bsz( x ) = ($LEN(x))#,           ! Get string length of object
  Len^Wsz( x ) = (TO^WSZ($LEN(x)))#;   ! Get word length of object
DEFINE
  Blankb(obj) = obj ':=' [" "] & obj FOR Len^Bsz(obj) - 1#, ! BYTE-objects only
  Blankw(obj) = obj ':=' "  "  & obj FOR Len^Wsz(obj) - 1#, ! WORD-objects only
  Zerob(obj)  = obj ':=' [0]   & obj FOR Len^Bsz(obj) - 1#, ! BYTE-objects only
  Zerow(obj)  = obj ':='  0    & obj FOR Len^Wsz(obj) - 1#; ! WORD-objects only
?PAGE"STATUS COLLECTION SERVER LITERAL AND DEFINES"
------------------------------------------------------------------------------
--
-- This section will first load all the ViewPoint Structure definitions
-- include in the file GTAL distributed with the ViewPoint Subvolume.
--
-- The first set of literals present the request codes that this server
-- has to support.
--
-- The second set of literals present the possible errors that can be
-- returned to the requesters.
--
-- The DEFINE Allocate^Item, will be use in procedure Server^Initialize
-- to initialize the Item^Array structure who contains the definitions and
-- initial parameters of all supported status items.
--
------------------------------------------------------------------------------
?NOLIST, SOURCE GTAL
?LIST
-- Request-Code Values
LITERAL  Z^Get^Version^Code          =    1;
LITERAL  Z^Scan^Item^Code            =    2;
LITERAL  Z^Next^Item^Code            =    3;
LITERAL  Z^Check^Item^Code           =    4;
LITERAL  Z^Sample^Item^Code          =    5;
-- Error-Code Values
LITERAL  Z^All^Ok^Code               =    0;
LITERAL  Zerr^Stat^Collect^Request   = 0206;
LITERAL  Zerr^Stat^No^Such^Type      = 0207;
LITERAL  Zerr^Stat^No^Such^Object    = 0208;
LITERAL  Zerr^Stat^End^Of^Types      = 0209;
LITERAL  Zerr^Stat^Cnfg^Incompatible = 0210;
LITERAL  Zerr^Stat^Object^Unavailable = 0211;
DEFINE   Allocate^Item ( Index, Description, Type^, Object^,
                         Value^For^100^, Use^Maximum^,
                         Reverse^Signify^, Scale^,
                         Low^Threshold^, High^Threshold^,
                         Low^Enabled^, High^Enabled^,
                         Value^Type^ ) =
                         @Item := @Item^Array[Index];
                         BLANKB( Item.Z^Item^Description );
                         Item.Z^Item^Description ':=' Description;
                         BLANKB( Item.Z^Item^Name );
                         Item.Z^Item^Name.Z^Item^Type ':=' Type^;
                         Item.Z^Item^Name.Z^Object^Name ':=' Object^;
                         Item.Z^Value^For^100 := Value^For^100^;
                         Item.Z^Use^Maximum := Use^Maximum^;
                         Item.Z^Reverse^Signify := Reverse^Signify^;
                         Item.Z^Scale := Scale^;
                         Item.Z^Low^Threshold := Low^Threshold^;
                         Item.Z^High^Threshold := High^Threshold^;
                         Item.Z^Low^Enabled := Low^Enabled^;
                         Item.Z^High^Enabled := High^Enabled^;
                         Item.Z^Value^Type ':=' Value^Type^
                         #;
```

```
?PAGE "STATUS COLLECTION SERVER GLOBAL VARIABLES AND DEFINITION"
------------------------------------------------------------------------------
--
-- This section allocate global variables and initialize them.
--
-- The Defined^Items define will be initialized to the number of item
-- types supported by this server.
--
-- STATUS ITEMS implemented by this server are defined in the table
-- Item^Array who contains an element for each implemented item.
-- The default values for the fields of each status item are provided
-- in this array.
--
-- Finally we will load all GUARDIAN Procedures Declarations from the
-- C00 EXTDECS0 file.
--
------------------------------------------------------------------------------
DEFINE  Current^Version = "C00"#; ! Current version of this server
DEFINE  Defined^Items   =  6#;    ! Number of Items supported by this server.
LITERAL Zvpt^Ipc^Len    =  4000;   ! Viewpoint Maximum Ipc Length
STRUCT .EXT Item^Array ( Zvpt^Status^Item^Def ) [ 1:Defined^Items ];
INT     Rcv^Fnum;                 ! $RECEIVE file number
INT     Num^Requesters      := 0; ! Number of requesters opening us
INT     Return^Error        := 0; ! This is use for the implementation of
INT     Return^Error^Detail := 0; ! error detection and reporting to the
                                  ! requester.
?NOLIST, SOURCE Extdecs0 ( ABEND, AWAITIO, CLOSE, CREATEPROCESSNAME,
?    CONVERTTIMESTAMP, DEVICEINFO, KEYPOSITION, FILEINFO,
?    FNAMEEXPAND, JULIANTIMESTAMP, MYPID, MYSYSTEMNUMBER, NEWPROCESS,
?    NUMIN, OPEN, PROCESSACCESSID, PROCESSINFO, PROCESSORSTATUS,
?    PROCESSORTYPE, PROCESSTIME,LOOKUPPROCESSNAME, PROGRAMFILENAME,
?    READ, READUPDATE, REPLY, SETMODE, SHIFTSTRING , WRITE, WRITEREAD,
?    COMPUTETIMESTAMP, STOP, DISKINFO, INITIALIZER )
?LIST
?PAGE "PROCEDURE SERVER^STARTUP"
PROC Server^Startup;
BEGIN
------------------------------------------------------------------------
--
--  This procedure is called before anything else happens.
--  It is responsible for processing any startup messages and prepare
--  the process for accepting messages on $RECEIVE.
--  Note the use of the INITIALIZER procedure who will take care of the
--  startup message protocol.
--  We will then open $RECEIVE for real work processing.
--
------------------------------------------------------------------------
INT     Rcv^Fname[0:11] := ["$RECEIVE", 8 * ["  "]];
INT     Rcv^Error;
LITERAL Want^Sysmsgs    = %040000;  ! Means we want to see OPEN, CLOSE Msgs.
LITERAL Receive^Depth   = %000001;  ! Means we want to Reply to Readupdate
  CALL INITIALIZER;
  CALL OPEN ( Rcv^Fname, Rcv^fnum, Want^Sysmsgs, Receive^Depth );
  IF <> THEN BEGIN
               CALL FILEINFO ( Rcv^Fnum, Rcv^Error );
               CALL ABEND;
             END;
  CALL SETMODE ( Rcv^Fnum, 36, 1 );
  IF <  THEN BEGIN
               CALL FILEINFO ( Rcv^Fnum,Rcv^Error );
               CALL ABEND;
            END;
END; ! Server^Startup
?PAGE "  SERVER^INITIALIZE"
PROC Server^Initialize;
BEGIN
------------------------------------------------------------------------------
--
--  This procedure is called before the first IPC request is received.
--  This is where a server typically does most of its initialization.
--  Here, we will initialize the global structure Item^Array with the
--  following objects supported by this server.
--  The Item pointer is used to help initialize all fields of the
--  Item^Array extended structure.
```

```
--
------------------------------------------------------------------------------
INT   .EXT Item ( Zvpt^Status^Item^Def );
  Allocate^Item ( 1,                ! index
                  "Process Time",   ! description
                  "PROCESS-TIME",   ! ITEM TYPE
                  "<Process-Name>", ! object name
                  100F,             ! value for 100
                  "N",              ! use maximum ?
                  "N",              ! reverse signify
                  0,                ! scale
                  0,                ! low threshold
                  90,               ! high threshold
                  "N", "Y",         ! low/high threshold enabled
                  "BT"              ! type (busy-timer or counter-value)
                  );
  Allocate^Item ( 2,                ! index
                  "Process Count",  ! description
                  "PROCESS-COUNT",  ! ITEM TYPE
                  "<Process-Name>", ! object name
                  100F,             ! value for 100
                  "Y",              ! use maximum ?
                  "N",              ! reverse signify
                  0,                ! scale
                  0,                ! low threshold
                  100,              ! high threshold
                  "N", "N",         ! low/high threshold enabled
                  "CV"              ! type (busy-timer or counter-value)
                  );
  Allocate^Item ( 3,                ! index
                  "File Name",      ! description
                  "FILE-USAGE",     ! ITEM TYPE
                  "<Filename>",     ! object name
                  100F,             ! value for 100
                  "Y",              ! use maximum ?
                  "N",              ! reverse signify
                  0,                ! scale
                  0,                ! low threshold
                  70,               ! high threshold
                  "N", "Y",         ! low/high threshold enabled
                  "CV"              ! type (busy-timer or counter-value)
                  );
  Allocate^Item ( 4,                ! index
                  "Disk Usage",     ! description
                  "DISK-USAGE",     ! ITEM TYPE
                  "<Disk Name>",    ! object name
                  100F,             ! value for 100
                  "Y",              ! use maximum ?
                  "N",              ! reverse signify
                  0,                ! scale
                  0,                ! low threshold
                  100,              ! high threshold
                  "N", "N",         ! low/high threshold enabled
                  "CV"              ! type (busy-timer or counter-value)
                  );
  Allocate^Item ( 5,                ! index
                  "Disk Largest Free Extent",  ! description
                  "DISK-LARGEST-FREE",  ! ITEM TYPE
                  "<Disk Name>",    ! object name
                  100F,             ! value for 100
                  "Y",              ! use maximum ?
                  "N",              ! reverse signify
                  0,                ! scale
                  0,                ! low threshold
                  100,              ! high threshold
                  "N", "N",         ! low/high threshold enabled
                  "CV"              ! type (busy-timer or counter-value)
                  );
  Allocate^Item ( 6,                ! index
                  "Disk Fragmentation",  ! description
                  "DISK-FRAGMENTATION",  ! ITEM TYPE
                  "<Disk Name>",    ! object name
                  100F,             ! value for 100
                  "Y",              ! use maximum ?
```

```
                  "N",            ! reverse signify
                  0,              ! scale
                  0,              ! low threshold
                  100,            ! high threshold
                  "N", "N",       ! low/high threshold enabled
                  "CV"            ! type (busy-timer or counter-value)
                  );
END;
?PAGE "SERVER^SHUTDOWN"
PROC Server^Shutdown;
BEGIN
------------------------------------------------------------------------------
--
--  This procedure is called right before the server stops.  A server
--  may do any relevant housecleaning here.
--
------------------------------------------------------------------------------
  CALL CLOSE ( Rcv^Fnum );
END;
?PAGE "FIND^ITEM^PROCEDURE"
INT PROC Find^Item^Type ( Item^Name, Index );
INT .EXT Item^Name ( Zvpt^Item^Name^Def ); ! Input:  Item to search.
INT     .Index;                           ! Output: Index of Item
BEGIN
------------------------------------------------------------------------------
--
-- Procedure searches allocated status item types for one matching the
-- type passed as a parameter.  If a match is found, the index of the
-- matching status item is set, and TRUE is returned.
--
------------------------------------------------------------------------------
LITERAL Item^Type^Len = Len^Bsz ( Item^Name.Z^Item^Type );
STRING .Item^Type [ 0:Item^Type^Len - 1 ];
  Item^Type ':=' Item^Name.Z^Item^Type FOR Item^Type^Len BYTES;
  CALL SHIFTSTRING ( Item^Type, Item^Type^Len, 0 );
  USE i;
  FOR i := 1 TO Defined^Items DO
    BEGIN
      IF Item^Array[i].Z^Item^Name.Z^Item^Type = Item^Type FOR
                                                 Item^Type^Len BYTES THEN
      BEGIN
        Index := i;
        RETURN TRUE;
      END;
    END;
  DROP i;
  RETURN FALSE;
END;  ! Find^item^type
?PAGE "CHECK^VALID^PROCESS^NAME"
INT  PROC Check^Valid^Process^Name ( Request, Reply );
INT .EXT  Request ( Zvpt^Check^Item^Request^Def ); ! Input
INT .EXT  Reply;                                   ! Output
BEGIN
------------------------------------------------------------------------
--
-- Procedure checks that the configured name for Processname is Valid.
-- First we check only for a valid process name
-- Later we could implement the process check for <CPU,PIN>
--
-- We do not require that the process is running.
-- The external form of the name should not be less than 2 characters ($a)
-- or greater than 14 (\remote7.$long).
-- The expanded internal form must have blank fill in subvol/name fields.
--
------------------------------------------------------------------------
INT    .Processname [ 0:Fname^Wsz - 1 ];      ! Internal form of name.
INT    .Temp^Fname  [ 0:Fname^Wsz - 1 ];      ! Temp file name.
STRUCT .Item^Name   ( Zvpt^Item^Name^Def );   ! Local structure.
INT     Length;                               ! Length of external name.
  Return^Error := 0;
  Return^Error^Detail := 0;
  Item^Name ':=' Request.Z^Status^Item.Z^Item^Name
                 FOR Len^Wsz ( Item^Name ) WORDS;
  CALL PROGRAMFILENAME ( Temp^Fname );
```

```
  Length := FNAMEEXPAND ( Item^Name.Z^Object^Name, Processname, Temp^Fname );
  IF Length < 2 OR Length > 14 THEN
    RETURN ( Zerr^Stat^No^Such^Object );
  Temp^Fname ':=' "  " & Temp^Fname FOR 7;
  IF Processname[4] <> Temp^Fname FOR 8 THEN
    RETURN ( Zerr^Stat^No^Such^Object );
  RETURN ( Z^All^Ok^Code );
END; ! Check^Process^Exist
?PAGE "CHECK^VALID^FILE^NAME"
INT  PROC Check^Valid^File^Name ( Request, Reply );
INT .EXT  Request ( Zvpt^Check^Item^Request^Def ); ! Input
INT .EXT  Reply;                                   ! Output
BEGIN
------------------------------------------------------------------------------
--
-- Procedure checks that the configured File name is Valid.
-- Right now, the file has to exist to pass the check, if this is a limitation
-- then you can change this code to verify only the file name format.
--
------------------------------------------------------------------------------
STRUCT .Item^Name          ( Zvpt^Item^Name^Def );
INT     Internal^File^Name [ 0:Fname^Wsz - 1 ];
INT     Temp^Fname         [ 0:Fname^Wsz - 1 ];
INT     Length;                          ! Length of external name.
INT     File^Error;
  Return^Error := 0;
  Return^Error^Detail :=0;
  Item^Name ':=' Request.Z^Status^Item.Z^Item^Name
                 FOR Len^Wsz ( Item^Name ) WORDS;
  CALL PROGRAMFILENAME ( Temp^Fname );
  Length := FNAMEEXPAND ( Item^Name.Z^Object^Name, Internal^File^Name,
                          Temp^Fname );
  IF Length = 0 THEN RETURN ( Zerr^Stat^No^Such^Object );
  CALL FILEINFO ( ,File^Error, Internal^File^Name );
  IF <> THEN
     BEGIN
       Return^Error^Detail := File^Error;
       RETURN ( Zerr^Stat^No^Such^Object );
    END;
  RETURN ( Z^All^Ok^Code );
END; ! Check^Valid^File^Name
?PAGE "CHECK^VALID^DISK^NAME"
INT  PROC Check^Valid^Disk^Name ( Request, Reply );
INT .EXT  Request ( Zvpt^Check^Item^Request^Def ); ! Input
INT .EXT  Reply;                                   ! Output
BEGIN
------------------------------------------------------------------------------
--
-- Procedure checks that the Disk Name is valid and existing on this system.
--
------------------------------------------------------------------------------
STRUCT .Item^Name ( Zvpt^Item^Name^Def );
INT    .Disk^Name [ 0:Fname^Wsz - 1 ];
INT     DevType;
INT     DevRecSize;
  Return^Error := 0;
  Return^Error^Detail :=0;
  Item^Name ':=' Request.Z^Status^Item.Z^Item^Name
                 FOR Len^Wsz ( Item^Name ) WORDS;
  Disk^Name ':=' "  " & Disk^Name FOR 11 WORDS;
  Disk^Name ':=' Item^Name.Z^Object^Name FOR 12 WORDS;
  CALL DEVICEINFO ( Disk^Name, DevType, DevRecSize );
  IF DevType.<4:9> <> 3 THEN RETURN ( Zerr^Stat^No^Such^Object )
                        ELSE RETURN ( Z^All^Ok^Code );
END; ! Check^Valid^Disk^Name
?PAGE "SAMPLE^PROCESS^BUSY"
INT  PROC Sample^Process^Busy ( Request, Reply );
INT .EXT  Request ( Zvpt^Sample^Item^Request^Def ); ! Input
INT .EXT  Reply   ( Zvpt^Sample^Item^Reply^Def );   ! Output
BEGIN
------------------------------------------------------------------------------
--
--  Computes cpu-busy-time for this process or process pair using the
--  PROCESSINFO call.
```

```
--
------------------------------------------------------------------------------
FIXED   Primary^Process^Time := 0F;
FIXED   Backup^Process^Time := 0F;
INT    .Processname [ 0:Fname^Wsz - 1 ];       ! internal form of name.
INT    .Temp^Fname  [ 0:Fname^Wsz - 1 ];       ! temp file name.
STRUCT .Item^Name   ( Zvpt^Item^Name^Def );
INT     Cpu^Pin;
INT    .PPD[0:8];
INT     Length;
  Zerow(Reply);
  Return^Error := 0;
  Return^Error^Detail := 0;
  Item^Name ':=' Request.Z^Item^Name
                 FOR Len^Wsz ( Item^Name ) WORDS;
  CALL PROGRAMFILENAME ( Temp^Fname );
  Length := FNAMEEXPAND ( Item^Name.Z^Object^Name, Processname, Temp^Fname );
  IF Length < 2 OR Length > 14
     THEN RETURN ( Zerr^Stat^No^Such^Object );
  Temp^Fname ':=' "  " & Temp^Fname FOR 7;
  IF Processname[4] <> Temp^Fname FOR 8
     THEN RETURN ( Zerr^Stat^No^Such^Object );
  PPD ':=' Processname FOR 9 WORDS;
  CALL LOOKUPPROCESSNAME (PPD);
  IF <> THEN RETURN ( Zerr^Stat^Object^Unavailable );
  IF PPD <> 0 THEN Cpu^Pin := PPD[3];
  IF ( Primary^Process^Time := Processtime ( Cpu^Pin ) ) < 0F
       THEN RETURN ( Zerr^Stat^Object^Unavailable )
       ELSE IF PPD[4] <> 0 THEN
       BEGIN
         Cpu^Pin := PPD[4];
         IF ( Backup^Process^Time := Processtime ( Cpu^Pin ) ) < 0F
         THEN Backup^Process^Time := 0F;
       END;
  Reply.Z^Status^Value := Primary^Process^Time + Backup^Process^Time;
  Reply.Z^Timestamp := Juliantimestamp;
  RETURN ( Z^All^Ok^Code );
END; ! Sample^Process^Busy
?PAGE "SAMPLE^PROCESS^COUNT"
INT  PROC Sample^Process^Count ( Request, Reply );
INT .EXT  Request ( Zvpt^Sample^Item^Request^Def ); ! Input
INT .EXT  Reply   ( Zvpt^Sample^Item^Reply^Def );   ! Output
BEGIN
------------------------------------------------------------------------------
--
-- Check to see if a process exist   Usefull      May be ...
--
------------------------------------------------------------------------------
STRUCT .Item^Name   ( Zvpt^Item^Name^Def );
INT    .Processname [ 0:Fname^Wsz - 1 ];      ! internal form of name.
INT    .Temp^Fname  [ 0:Fname^Wsz - 1 ];      ! temp file name.
INT     Cpu^Pin;
INT    .PPD[0:8];
INT     Length;
  Zerow ( Reply );
  Return^Error := 0;
  Return^Error^Detail := 0;
  Item^Name ':=' Request.Z^Item^Name
                 FOR Len^Wsz ( Item^Name ) WORDS;
  CALL PROGRAMFILENAME ( Temp^Fname );
  Length := FNAMEEXPAND ( Item^Name.Z^Object^Name, Processname, Temp^Fname );
  IF Length < 2 OR Length > 14
     THEN RETURN ( Zerr^Stat^No^Such^Object );
  Temp^Fname ':=' "  " & Temp^Fname FOR 7;
  IF Processname[4] <> Temp^Fname FOR 8
     THEN RETURN ( Zerr^Stat^No^Such^Object );
  PPD ':=' Processname FOR 9 WORDS;
  CALL LOOKUPPROCESSNAME ( PPD );
  IF <> THEN RETURN ( Zerr^Stat^Object^Unavailable );
  Reply.Z^Maximum^Value := 1F;
  Reply.Z^Status^Value := 1F;
  Reply.Z^Timestamp := Juliantimestamp;
  RETURN ( Z^All^Ok^Code );
END; ! Sample^Process^Count
```

```
?PAGE "SAMPLE^FILE^USAGE"
INT  PROC Sample^File^Usage ( Request, Reply );
INT .EXT  Request ( Zvpt^Sample^Item^Request^Def ); ! Input
INT .EXT  Reply   ( Zvpt^Sample^Item^Reply^Def );   ! Output
BEGIN
------------------------------------------------------------------------------
--
-- Compute File Usage for this file ......
--
------------------------------------------------------------------------------
INT(32) Current^File^Size;
INT(32) Maximum^File^Size;
STRUCT .Item^Name          ( Zvpt^Item^Name^Def );
INT     Temp^Fname         [ 0:Fname^Wsz - 1 ];
INT     Internal^File^Name [ 0:Fname^Wsz - 1 ];
INT     Length;
INT     File^error;
  Zerow ( Reply );
  Return^Error := 0;
  Return^Error^Detail := 0;
  Item^Name ':=' Request.Z^Item^Name FOR Len^Wsz ( Item^Name ) WORDS;
  CALL PROGRAMFILENAME ( Temp^Fname );
  Length := FNAMEEXPAND ( Item^Name.Z^Object^Name, Internal^File^Name,
                          Temp^Fname );
  IF Length = 0 THEN RETURN ( Zerr^Stat^No^Such^Object );
  CALL FILEINFO ( , File^Error, Internal^File^Name,,,, Current^File^Size,
                  ,,,,,,,,,, Maximum^File^Size );
  IF <> THEN
  BEGIN
    Return^Error^Detail := File^Error;
    RETURN ( Zerr^Stat^Object^Unavailable );
  END ELSE
  BEGIN
    Reply.Z^Maximum^Value :=  $DFIX ( Maximum^File^Size, 0);
    Reply.Z^Status^Value  :=  $DFIX ( Current^File^Size, 0);
    Reply.Z^Timestamp := Juliantimestamp;
   END;
   RETURN ( Z^All^Ok^Code );
END; ! Sample^File^Usage
?PAGE "SAMPLE^DISK^USAGE"
INT  PROC Sample^Disk^Usage ( Request, Reply );
INT .EXT  Request ( Zvpt^Sample^Item^Request^Def ); ! Input
INT .EXT  Reply   ( Zvpt^Sample^Item^Reply^Def );   ! Output
BEGIN
------------------------------------------------------------------------------
--
--  Compute Disk Usage or using the INVERSE SIGNIFIANCE flag Disk Free !!!!
--
------------------------------------------------------------------------------
STRUCT  .Item^Name ( Zvpt^Item^Name^Def );
INT     .Disk^Name [ 0:Fname^wsz - 1 ];
INT      DevType;
INT      DevRecSize;
INT(32)  Capacity;
INT(32)  Free;
INT(32)  Numfrag;
INT(32)  Biggest;
  Zerow ( Reply );
  Return^Error := 0;
  Return^Error^Detail := 0;
  Item^Name ':=' Request.Z^Item^Name FOR Len^Wsz ( Item^Name ) WORDS;
  Disk^Name ':=' "  " & Disk^Name FOR 11 WORDS;
  Disk^Name ':=' Item^Name.Z^Object^Name FOR 24 BYTES;
  CALL DEVICEINFO ( Disk^Name, DevType, DevRecSize );
  IF DevType.<4:9> <> 3 THEN RETURN ( Zerr^Stat^No^Such^Object );
  Return^Error^Detail := DISKINFO ( Disk^Name, Capacity,
                                    Free, Numfrag, Biggest );
  IF Return^Error^Detail <> 0 THEN
  BEGIN
    RETURN ( Zerr^Stat^Object^Unavailable );
  END;
  Reply.Z^Maximum^Value :=  $DFIX ( Capacity * 2048D , 0 );
  Reply.Z^Status^Value  :=  $DFIX ( ( Capacity * 2048D - Free * 2048D ) , 0 );
  Reply.Z^Timestamp := Juliantimestamp;
```

```
  RETURN ( Z^All^Ok^Code );
END; ! Sample^Disk^Usage
?PAGE "SAMPLE^DISK^LARGEST^FREE"
INT  PROC Sample^Disk^Largest^Free ( Request, Reply );
INT .EXT  Request ( Zvpt^Sample^Item^Request^Def ); ! Input
INT .EXT  Reply   ( Zvpt^Sample^Item^Reply^Def );   ! Output
BEGIN
------------------------------------------------------------------------------
--
-- Compute the disk largest free extent available for usage
--
------------------------------------------------------------------------------
STRUCT .Item^Name ( Zvpt^Item^Name^Def );
INT    .Disk^Name [ 0:Fname^Wsz - 1 ];
INT     DevType;
INT     DevRecSize;
INT(32) Capacity;
INT(32) Free;
INT(32) Numfrag;
INT(32) Biggest;
  Zerow ( Reply );
  Return^Error := 0;
  Return^Error^Detail := 0;
  Item^Name ':=' Request.Z^Item^Name FOR Len^Wsz ( Item^Name ) WORDS;
  Disk^Name ':=' "  " & Disk^Name FOR 11 WORDS;
  Disk^Name ':=' Item^Name.Z^Object^Name FOR 24 BYTES;
  CALL DEVICEINFO ( Disk^Name, DevType, DevRecSize );
  IF DevType.<4:9> <> 3 THEN RETURN ( Zerr^Stat^No^Such^Object );
  Return^Error^Detail := DISKINFO ( Disk^Name, Capacity,
                                    Free, Numfrag, Biggest );
  IF Return^Error^Detail <> 0 THEN
  BEGIN
    RETURN ( Zerr^Stat^Object^Unavailable );
  END;
  Reply.Z^Maximum^Value :=  $DFIX ( Capacity * 2048D , 0 );
  Reply.Z^Status^Value  :=  $DFIX ( Biggest * 2048D, 0 );
  Reply.Z^Timestamp := Juliantimestamp;
  RETURN ( Z^All^Ok^Code );
END; ! Sample^Disk^Largest^Free
?PAGE "SAMPLE^DISK^FRAGMENTATION"
INT  PROC Sample^Disk^Fragmentation ( Request, Reply );
INT .EXT  Request ( Zvpt^Sample^Item^Request^Def ); ! Input
INT .EXT  Reply   ( Zvpt^Sample^Item^Reply^Def );   ! Output
BEGIN
------------------------------------------------------------------------------
--
-- Compute the number of fragments on this disk.
--
------------------------------------------------------------------------------
STRUCT .Item^Name ( Zvpt^Item^Name^Def );
INT    .Disk^Name [ 0:Fname^Wsz - 1 ];
INT     DevType;
INT     DevRecSize;
INT(32) Capacity;
INT(32) Free;
INT(32) Numfragment;
INT(32) Biggest;
  Zerow ( Reply );
  Return^Error := 0;
  Return^Error^Detail := 0;
  Item^Name ':=' Request.Z^Item^Name FOR Len^Wsz ( Item^Name ) WORDS;
  Disk^Name ':=' "  " & Disk^Name FOR 11 WORDS;
  Disk^Name ':=' Item^Name.Z^Object^Name FOR 24 BYTES;
  CALL DEVICEINFO ( Disk^Name, DevType, DevRecSize );
  IF DevType.<4:9> <> 3 THEN RETURN ( Zerr^Stat^No^Such^Object );
  Return^Error^Detail := DISKINFO ( Disk^Name, Capacity,
                                    Free, Numfragment, Biggest );
  IF Return^Error^Detail <> 0 THEN
  BEGIN
    RETURN ( Zerr^Stat^Object^Unavailable );
  END;
  Reply.Z^Maximum^Value :=  $DFIX ( Capacity * 2048D , 0 );
  Reply.Z^Status^Value  :=  $DFIX ( Numfragment, 0 );
  Reply.Z^Timestamp := Juliantimestamp;
```

```
  RETURN ( Z^All^Ok^Code );
END; ! Sample^Disk^Fragmentation
?PAGE "SCAN ITEM"
INT  PROC Scan^Item ( Request, Request^Len, Reply, Reply^Len );
INT .EXT  Request ( Zvpt^Scan^Item^Request^Def ); ! Input
INT .EXT  Request^Len;                            ! Length of data in 'Request'
INT .EXT  Reply ( Zvpt^Scan^Item^Reply^Def );     ! Output
INT .EXT  Reply^Len;                              ! Length of data in 'Reply'.
BEGIN
------------------------------------------------------------------------
--
-- Scan^Item procedure searches the defined items in this server for one
-- matching the requested item^type.  If the requested item^type is blank
-- then the first defined type is returned.
-- The item index is included in the reply to allow "Scan^Next" to work.
-- Defined item indexes start at 1 and go to "defined^items".
--
------------------------------------------------------------------------
STRUCT .Blank^Name ( Zvpt^Item^Name^Def );
INT     Item^Index;
INT     Object^Index;
  Blankw ( Blank^Name );
  Return^Error := 0;
  IF Request.Z^Item^Name.Z^Item^Type = Blank^Name.Z^Item^Type FOR
                            Len^Wsz ( Blank^Name.Z^Item^Type ) THEN
  Item^Index := 1
  ELSE
  BEGIN
    IF NOT Find^Item^Type ( Request.Z^Item^Name, Item^Index ) THEN
      RETURN ( Zerr^Stat^No^Such^Type );
  END;
  Reply.Z^Status^Item ':=' Item^Array [ Item^Index ] FOR
                           Len^Bsz ( Reply.Z^Status^Item ) BYTES;
  Reply.Z^Item^Index := Item^Index;
  Reply^Len := Len^Bsz ( Zvpt^Scan^Item^Reply^Def );
  RETURN ( Z^All^Ok^Code );
END;  ! Scan^Item.
?PAGE "SCAN NEXT ITEM"
INT  PROC Scan^Next^Item ( Request, Request^Len, Reply, Reply^Len );
INT .EXT  Request ( Zvpt^Next^Item^Request^Def ); ! Input
INT .EXT  Request^Len;                            ! Length of data in 'Request'
INT .EXT  Reply   ( Zvpt^Next^Item^Reply^Def );   ! Output
INT .EXT  Reply^Len;                              ! Length of data in 'Reply'.
BEGIN
------------------------------------------------------------------------
--
-- Scan^Next^Item procedure is used after Scan^Next to retrieve the
-- next item in the list of defined items.  The index value is incremented
-- and the status item to which the new index value corresponds is returned.
-- The new index value is returned with the status item (for subsequent
-- calls to "scan^next").
-- If the new (incremented) index exceeds defined^items then Item^Index
-- is set back to 1 and so we start back at the beginning of the list.
--
------------------------------------------------------------------------
INT Item^Index;
INT Object^Index;
  Return^Error := 0;
  Item^Index := Request.Z^Item^Index + 1;
  IF Item^Index > Defined^Items THEN Item^Index := 1;
  Reply.Z^Status^Item ':=' Item^Array [ Item^Index ] FOR
                           Len^Bsz ( Reply.Z^Status^Item ) BYTES;
  Reply.Z^Item^Index := Item^Index;
  Reply^Len := Len^Bsz ( Zvpt^Next^Item^Reply^Def );
  RETURN ( Z^All^Ok^Code );
END;  ! Scan^Next^Item.
?PAGE "CHECK ITEM"
INT  PROC Check^Item ( Request, Request^Len, Reply, Reply^Len );
INT .EXT  Request ( Zvpt^Check^Item^Request^Def ); ! Input
INT .EXT  Request^Len;                              ! Length of data in 'Request
INT .EXT  Reply   ( Zvpt^Check^Item^Reply^Def );   ! Output
INT .EXT  Reply^Len;                                ! Length of data in 'Reply'.
BEGIN
------------------------------------------------------------------------------
```

```
--
-- Check^Item implements the checking of a defined items. The defined item
-- index is determined by a search of the list of items. The actual checking
-- is then performed by a procedure Check^Valid^........
--
------------------------------------------------------------------------------
INT Index;                  ! Index of item in item^array.
  Return^Error := 0;
  IF Request^Len < Len^Bsz ( Zvpt^Check^Item^Request^Def ) THEN
    RETURN ( Zerr^Stat^Collect^Request );
  IF NOT Find^Item^Type ( Request.Z^Status^Item.Z^Item^Name, Index ) THEN
    RETURN ( Zerr^Stat^No^Such^Type );
  Reply.Z^Status^Item ':=' Request.Z^Status^Item FOR 1 ELEMENTS;
  CASE Index OF
  BEGIN
    1 -> Return^Error := Check^Valid^Process^Name ( Request, Reply );
    2 -> Return^Error := Check^Valid^Process^Name ( Request, Reply );
    3 -> Return^Error := Check^Valid^File^Name    ( Request, Reply );
    4 -> Return^Error := Check^Valid^Disk^Name    ( Request, Reply );
    5 -> Return^Error := Check^Valid^Disk^Name    ( Request, Reply );
    6 -> Return^Error := Check^Valid^Disk^Name    ( Request, Reply );
    OTHERWISE ->  RETURN ( Zerr^Stat^No^Such^Type );
  END; ! case on item index
  IF ( Return^Error = 0 ) THEN
  BEGIN
    Reply^Len := Len^Bsz ( Zvpt^Check^Item^Reply^Def );
    RETURN ( Return^Error );
  END
  ELSE RETURN ( Return^Error );
END;  ! Check^Item.
?PAGE "SAMPLE ITEM"
INT PROC Sample^Item ( Request, Request^Len, Reply, Reply^Len );
INT .EXT Request ( Zvpt^Sample^Item^Request^Def ); ! Input
INT .EXT Request^Len;                              ! Length of data in 'Request
INT .EXT Reply   ( Zvpt^Sample^Item^Reply^Def );   ! Output
INT .EXT Reply^Len;                                ! Length of data in 'Reply'
BEGIN
------------------------------------------------------------------------
--
-- Procedure implements the sample of a defined item.  The defined item
-- index is determined by a search of the list of items.  The actual
-- sample is then performed by a procedures Sample^.....
--
------------------------------------------------------------------------
INT Index;                  ! index of item in item^array.
  Return^Error := 0;
  IF Request^Len < Len^Bsz ( Zvpt^Sample^Item^Request^Def ) THEN
    RETURN ( Zerr^Stat^Collect^Request );
  IF NOT Find^item^type ( Request.Z^Item^Name, Index ) THEN
    RETURN ( Zerr^Stat^No^Such^Type );
  CASE Index OF
  BEGIN
    1 -> Return^Error := Sample^Process^Busy      ( Request, Reply );
    2 -> Return^Error := Sample^Process^Count     ( Request, Reply );
    3 -> Return^Error := Sample^File^Usage        ( Request, Reply );
    4 -> Return^Error := Sample^Disk^Usage        ( Request, Reply );
    5 -> Return^Error := Sample^Disk^Largest^Free ( Request, Reply );
    6 -> Return^Error := Sample^Disk^Fragmentation ( Request, Reply );
    OTHERWISE -> CALL ABEND;
  END; ! case on item index
  IF ( Return^Error = 0 ) THEN
  BEGIN
    Reply^Len := Len^Bsz ( Zvpt^Sample^Item^Reply^Def );
    RETURN ( Return^Error );
  END
  ELSE RETURN ( Return^Error );
END; ! Sample^Item.
?PAGE "SERVER PERFORM REQUEST"
PROC Server^Perform^Request( Request^Buffer, Request^Len,
                             Reply^Buffer,   Reply^Len );
INT .EXT        Request^Buffer;  ! IN  The request to process.
INT .EXT        Request^Len;     ! IN  Length of data in 'RequestBuffer'.
INT .EXT        Reply^Buffer;    ! OUT Reply message to the request.
INT .EXT        Reply^Len;       ! OUT Length of data in 'ReplyBuffer'.
```

```
BEGIN
------------------------------------------------------------------------------
--
--  Perform the processing necessary to perform the request contained in
--  'RequestBuffer'.  Formulate a reply in 'ReplyBuffer'.
--
------------------------------------------------------------------------------
INT .EXT Request^Header ( Zvpt^Request^Header^Def ) = Request^Buffer;
INT .EXT Reply^Header   ( Zvpt^Reply^Header^Def )   = Reply^Buffer;
INT .EXT Sub^Request^Buffer := @Request^Buffer[Len^Wsz(Zvpt^Request^Header^Def)];
INT .EXT Sub^Reply^Buffer   := @Reply^Buffer[Len^Wsz(Zvpt^Reply^Header^Def)];
INT      Sub^Request^Len;
INT      Sub^Reply^Len := 0;
INT      Request^Code := Request^Header.Z^Request^code;
INT      Error := 0;
  Return^Error := 0;
  Return^Error^Detail := 0;
  Sub^Request^Len := Request^Len - Len^Bsz ( Zvpt^Request^Header^Def );
  Sub^Reply^Len := 0;
  IF Sub^Request^Len < 0 THEN Return^Error := ( Zerr^Stat^Collect^Request )
  ELSE
  BEGIN
   CASE Request^code OF
   BEGIN
     Z^Get^Version^Code -> ;
     Z^Scan^Item^Code   -> Return^Error := Scan^Item      ( Sub^Request^Buffer,
                                                           Sub^Request^Len,
                                                           Sub^Reply^Buffer,
                                                           Sub^Reply^Len);
     Z^Next^Item^Code   -> Return^Error := Scan^Next^Item ( Sub^Request^Buffer,
                                                           Sub^Request^Len,
                                                           Sub^Reply^Buffer,
                                                           Sub^Reply^Len );
     Z^Check^Item^Code  -> Return^Error := Check^Item     ( Sub^Request^Buffer,
                                                           Sub^Request^Len,
                                                           Sub^Reply^Buffer,
                                                           Sub^Reply^Len );
     Z^Sample^Item^Code -> Return^Error := Sample^Item    ( Sub^Request^Buffer,
                                                           Sub^Request^Len,
                                                           Sub^Reply^Buffer,
                                                           Sub^Reply^Len );
     OTHERWISE          -> Return^Error := Zerr^Stat^Collect^Request;
   END;  -- case.
  END;    -- Begin case.
  IF Return^Error = 0 THEN
    Reply^Header.Z^Ipc^Hdr.Z^Pw^Reply^Code := 0
  ELSE
  BEGIN
    Reply^Header.Z^Ipc^Hdr.Z^Ipc^Retn^Code := Return^Error;
    Reply^Header.Z^Ipc^Hdr.Z^Ipc^Retn^Code^Detail := Return^Error^Detail;
    Reply^Header.Z^Ipc^Hdr.Z^Pw^Reply^Code := 3;
  END; -- error caught.
  Reply^Header.Z^Ipc^Hdr.Z^Version^Code ':=' [Current^Version];
  Reply^Len := Sub^Reply^Len + Len^Bsz ( Reply^Header );
END;  ! Server^Perform^Request.
?PAGE "READ RECEIVE MESSAGE"
PROC  Read^Receive^Message ( Buffer, Actual^Length, Status ) VARIABLE;
INT  .Buffer;                ! OUTPUT: The message is returned here.
                             !         The buffer should be at least
                             !         'Max^Request + 1' bytes long.
INT  .Actual^Length;         ! OUTPUT: Actual length of message returned.
INT  .Status;                ! OUTPUT: Type of message read. One of:
                             !          Recv^User^Msg = 0
                             !          Recv^System^Msg^From^User = 1
                             !          Recv^System^Msg^From^System = 2
                             !          Recv^End^Of^File = 3.
BEGIN
------------------------------------------------------------------------------
--
--  This procedure will read one message from $RECEIVE.  It will then
--  verify if it is a system message and if so will check for OPEN messages
--  and CLOSE messages to update the Num^Requesters counter. If this is any
--  other system message we do not do anything with them. If it is a user
--  message we will set the Status variable to 1.
```

```
--
------------------------------------------------------------------------------
INT     Error := 0;
LITERAL System^Message = 6;
  CALL READUPDATE ( Rcv^Fnum, Buffer, Zvpt^Ipc^Len, Actual^Length );
  IF  > THEN
  BEGIN
    CALL Fileinfo ( Rcv^Fnum, Error );
    IF Error = System^Message
    THEN
    BEGIN
      CASE $ABS ( Buffer[0] ) - 30 OF
      BEGIN
   !0!  Num^Requesters := Num^Requesters + 1; ! -30 Open Msg from Requester
   !1!  Num^Requesters := Num^Requesters - 1; ! -31 Close Msg from Requester
        OTHERWISE;
      END;    !End of Case
    END       !End of System^Message
  END        !End of IF > THEN
  ELSE IF Error <> 0 THEN CALL ABEND;
  IF Error = 0 THEN Status := Recv^User^Msg
  ELSE IF ( Error = 6 AND Num^Requesters = 0 ) THEN Status:= Recv^End^of^file
  ELSE IF ( Error = 6 AND Num^Requesters > 0 )
       THEN Status := Recv^System^Msg^From^User;
END;  ! PROC Read^Receive^Message
?PAGE "REPLY^RECEIVE^MESSAGE"
PROC Reply^Receive^Message ( Buffer, Length ) VARIABLE;
INT .Buffer;      ! INPUT,OPTIONAL:  Text of the reply.  Omit
                  !                  only if 'Length' is zero.
INT  Length;      ! INPUT,OPTIONAL:  Length of the reply, in
                  !                  bytes.  Default is zero.
BEGIN
------------------------------------------------------------------------------!
--
-- This procedure will reply to a $RECEIVE message that was read by            !
-- 'Read^Receive^Message'.                                                     !
--
------------------------------------------------------------------------------!
  IF NOT $PARAM ( Buffer ) THEN @Buffer := 0;
  IF NOT $PARAM ( Length ) THEN Length := 0;
  CALL REPLY ( Buffer, Length );
END;  ! PROC Reply^Receive^Message
?PAGE "SERVER PROCESSING"
PROC Server^Processing;
BEGIN
------------------------------------------------------------------------------
--
--  This routine is called after all initialization is complete.
--  It will execute the server's main processing loop.
--
--  The implementation of this routine is appropriate for servers
--  that are single-threaded and are never interested in processing
--  system messages.  For each user request, 'Server^PerformRequest'
--  is called to execute the request.
--
------------------------------------------------------------------------------
INT .Request^Buf [0:Zvpt^Ipc^Len]; ! Request buffer
INT .Reply^Buf   [0:Zvpt^Ipc^Len]; ! Reply buffer
INT  Request^Len;                  ! Length of request
INT  Reply^Len;                    ! Length of reply
INT  Status;                       ! Status parameters of Read^Receive^Message
  WHILE 1 DO
  BEGIN
    CALL Read^Receive^Message ( Request^Buf, Request^Len, Status );
      IF Status = Recv^User^Msg THEN
         CALL Server^Perform^Request ( Request^Buf, Request^Len,
                                       Reply^Buf,   Reply^Len )
      ELSE IF Status = Recv^System^Msg^From^User THEN Reply^Len := 0
      ELSE IF Status = Recv^End^Of^File THEN RETURN;
    CALL Reply^Receive^Message ( Reply^Buf, Reply^Len );
  END;
END;  ! PROC Server^Processing
?PAGE "STATUS SERVER PRODUCT VERSION PROCEDURE T2001C00^03FEB88"
PROC T2001C00^03FEB88;
```

```
BEGIN
------------------------------------------------------------------------------
--
--  This version procedure is very nice, since we can list it using the VPROC
--  tool and then know all the time the server version we are dealing with.
--
------------------------------------------------------------------------------
END;
?PAGE "STATUS MAIN PROCEDURE"
PROC Status MAIN;
BEGIN
------------------------------------------------------------------------------
--
--  This is the main procedure invoking this string of procedures.
--  After Server^Startup processing, we do Server^Initialize and then go to
--  Server^Processing who is the main loop and stay there until all requester
--  close us. Finally we will terminate the program with Server^Shutdown.
--
------------------------------------------------------------------------------
  CALL Server^Startup;
  CALL Server^Initialize;
  CALL Server^Processing;
  CALL Server^Shutdown;
END;
```

E

Supplemental Information for D-Series Systems

This appendix tells you how to use ViewPoint on a D-series system. It consists of two major subsections: The first is a brief overview of D-series operating system features applicable to ViewPoint; the second is a description of the specific D-series ViewPoint enhancements.

You also might find the following manuals useful:

- *Introduction to D-Series Systems*
- *D-Series System Migration Planning Guide*
- *Guardian Application Conversion Guide*
- *TACL Reference Manual*
- *EMS Manual*

D-Series Operating System Features Applicable to ViewPoint

The D-series operating system expands many previous limits to allow full use of advances in Compaq system architecture. This subsection is a brief summary of some of the operating system features that affect ViewPoint. Refer to additional D-series documentation (see list above) for more background.

The next subsection, "ViewPoint Enhancements," describes the specific D-series enhancements to ViewPoint. If you are already familiar with D-series operating system features, you might prefer to go directly to this subsection.

Larger CPU and PIN Values

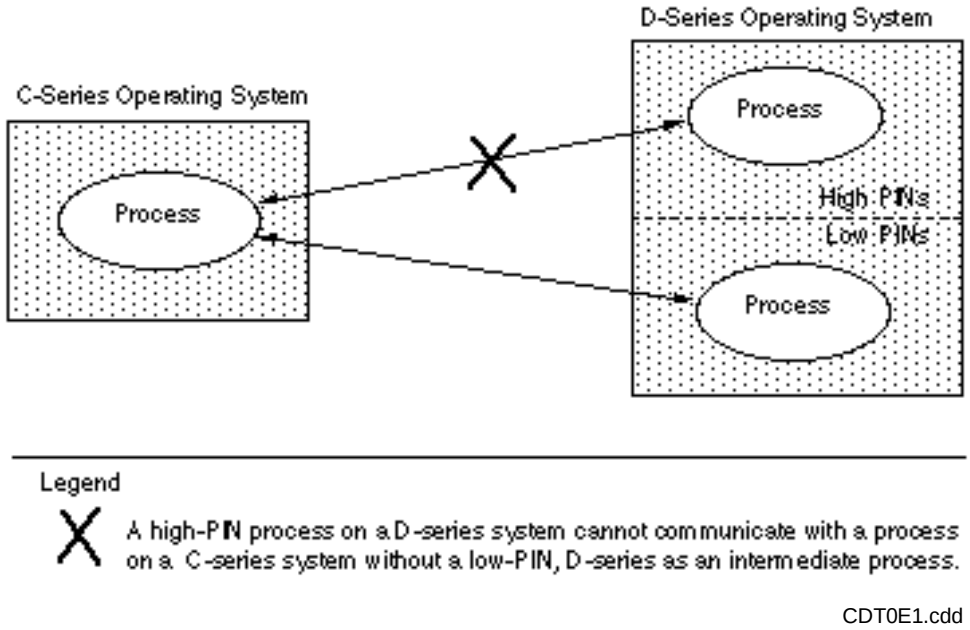
The valid CPU number range is 0 through 15 for both C-series and D-series systems. The CPU value is a maximum of 5 digits for D-series operating systems. This enhancement accommodates future expansion.

The range of process identification number (PIN) values for D-series systems is now 0 through the maximum number of processes allowed by the system and by resources on that system. Low PINs, 0 through 254, are applicable to both C-series and D-series operating systems. The high PINs, 256 and greater, are used on D-series operating systems. PIN 255 is reserved for use with a synthetic process ID and is valid for both C-series and D-series operating systems. The D-series systems have a PIN value maximum of 5 digits.

Communication Between Low-PIN and High-PIN Processes

A high-PIN process on a D-series system cannot communicate with a process on a C-series system without a low-PIN, D-series process, as an intermediate process. However, low-PIN processes on a D-series system can communicate directly with processes on a C-series system. On a D-series system, processes can communicate with each other, whether they are low-PIN or high-PIN processes. Figure E-1 illustrates these concepts.

Figure E-1. Communication Between low-PIN and high-PIN Processes



Note that in order for a process to run at high PIN, it must have certain characteristics. For example, you might not be able to take a C-series program and run it on a D-series system at high PIN. The program might need to be converted first. In addition, high-PIN processes and low-PIN processes must meet certain requirements to communicate with each other. See the *Guardian Application Conversion Guide* and the *Introduction to D-Series Systems* for more information.

Process Descriptor

A process descriptor is a variable-length string that specifies either a named or unnamed process (or a named process pair). In an EMS filter, to extract a process name or system name from an EMS token, use the DECOMPOSE function. The DECOMPOSE function can obtain this information from the process descriptor.

For more information on the process descriptor, see the *Guardian Application Conversion Guide*. For information on the DECOMPOSE function, see the *EMS Manual* section describing D-series changes.

ViewPoint Enhancements

This section describes specific D-series enhancements to ViewPoint. The following are covered:

- New D-Series Subject Tokens Supported
- Larger CPU/Process ID Field in the Event Configuration Screen
- Release-Specific Information in the Event Detail Screen
- DP Command HIGHPIN Option
- DP Command EXTSWAP Option
- ViewPoint Operation at a Low PIN
- Modifications to the Default Alternate-Events Filter

New D-Series Subject Tokens Supported

The D-series enhancements include two additional unsigned integer subject tokens: ZEMS-TKN-LDEVNUMBER and ZEMS-TKN-XSYSPID. The Subject field of the Last Events screen is described in the subsection, “Last Events Screen” in Section 3, Definition of ViewPoint Screens.

Larger CPU/Process ID Field in Event Configuration Screen

This field can contain a process ID consisting of cpu,pin or a process name, or it can contain just a CPU number This field is 11 digits for D-series systems. D-series systems support PIN values of 5 digits and CPU values of 5 digits. For more information, see the previous subsection, “Larger CPU and PIN Values.” The CPU/Process ID field of the Event Configuration screen is described in Section 3, Definition of ViewPoint Screens.

Release-Specific Information in Event Detail Screen

D-series messages can contain release-specific information. For example, the message can describe two different meanings for C-series and D-series operating systems. See the *Operator Messages: DSM Display Format* manual for more details. The Event Detail screen is described in Section 3, “Definition of ViewPoint Screens.”

DP Command HIGHPIN Option

The Define Process (DP) command has a HIGHPIN ON/OFF option for D-series operating systems. This option has several uses; for example, it allows you to run a high-PIN process at low PIN. The DP command is described in Section 2, Using ViewPoint and in Section 4, Process Definition Commands.

Following is an example of using this option at the TACL prompt to run the process APP at low PIN:

```
TACL> DP APP /HIGHPIN OFF/
```

If HIGHPIN ON is specified, the process runs at high PIN if it has been converted to run at high PIN. However, if the process has not been converted to run at high PIN, it runs at low PIN. The HIGHPIN option default is ON.

Whether a process is designed to run at high or low PIN depends upon the HIGHPIN object-file attribute. This attribute can be set either during compilation using a compiler directive or after compilation using the BINDER program.

See the previous subsection, "Communication Between Low-PIN and High-PIN Processes," for further information on PINs. The *Introduction to D-Series Systems* and the *Guardian Application Conversion Guide* provide more detailed background. The HIGHPIN option is also explained in the *TACL Reference Manual* section describing D-series features. HIGHPIN is a RUN command option.

DP Command EXTSWAP Option

The EXTSWAP option specifies the volume and, optionally, the file name that is used for memory swaps of the process' extended data, if any.

Following is an example of using this option at the TACL prompt:

```
TACL> DP APP /EXTSWAP EXT/
```

If the file EXT exists, it is used for swaps of the extended data segment during the execution of APP. If the file does not exist, it is created and used for swaps. No volume or subvolume was specified in the above example, so the system uses the =_DEFAULTS DEFINE VOLUME settings for these defaults.

If a disk device is specified for a file name, a temporary file is created on the specified disk device. If no file name or disk device is specified, the system uses the =_DEFAULTS DEFINE SWAP settings as the default and creates a temporary file. If there is no SWAP default, then the system places the temporary file wherever it chooses.

The EXTSWAP option is described in the *TACL Reference Manual* section describing D-series changes. It is a RUN command option.

Modifications to Default Alternate-Events Filter

Figure E-2 (later in this section) is a printout of the default alternate-events filter supplied with D-series ViewPoint. Following are the changes to the C21 default filter with the release of the D-series operating system. For more information, see Section 6, Customizing ViewPoint.

- The D-series token ZEMS-TKN-XSENDERID or ZEMS-TKN-XSENDERID-PD replaces C-series token ZEMS-TKN-SENDERID.
- CPU and PIN values are now obtained from the token ZEMS-TKN-XSENDERID or ZEMS-TKN-XSENDERID-PD instead of from the ZEMS-TKN-SYSPID and ZEMS-TKN-CRTPID tokens.
- The DECOMPOSE function is used to extract a process name or system name from a token. DECOMPOSE extracts this information from the process descriptor as described previously in the Section, "Process Descriptor."

Following are the four portions of the filter that have changed:

Test for Matching System Name

```
-- Test for matching system name (if provided).
if ( not tokenpresent(zvpt^fltr^systemname) or
     zvpt^fltr^systemname =
     decompose(ssid(zems^val^ssid,
                    zems^tkn^xsenderid^pd),system name)
   ) and
```

Test for Matching CPU

```
-- Test for matching cpu (if provided).
   ( not tokenpresent(zvpt^fltr^cpu) or
     zvpt^fltr^cpu =
     ssid(zems^val^ssid,
          zems^tkn^xsenderid).zems^ddl^xsenderid:zcpu
   ) and
```

Test for Matching PIN

```
-- Test for matching pin (if provided).
   ( not tokenpresent(zvpt^fltr^pin) or
     zvpt^fltr^pin =
     ssid(zems^val^ssid,
          zems^tkn^xsenderid).zems^ddl^xsenderid:zpin
   ) and
```

Test for Matching Process Name

D-Series

```
-- Test for matching process name (if provided).
   ( not tokenpresent(zvpt^fltr^processname) or
     decompose(zvpt^fltr^processname,
               destination name, name part) =
     decompose(ssid(zems^val^ssid,
               zems^tkn^xsenderid^pd),
               destination name,name part)
   ) and
```

Extraneous Parts of Filter

The following are obsolete for the D-series filter:

```
[#DEF fname32 STRUCT
   BEGIN
   CHAR  systemname (0:7);
   FNAME name;
   UINT  number (0:11) REDEFINES name;
```

```
      END;
   ]
   [#def zspi_ddl_crtpid STRUCT
      BEGIN
      BYTE Z_PROCNAME (0:5);
      BYTE Z_CPU;
      BYTE Z_PIN;
      END;
   ]
```

If you have customized filters, you might have to modify them. For more information, see the *EMS Manual* section describing D-series features.

Figure E-2. D-Series Default Alternate-Events Filter (page 1 of 4)

```
------------------------------------------------------------
-- ViewPoint default alternate event filter.
-- This filter processes parameters which may be specified
-- from the second page of the Alternate Event
-- Configuration screen.  ZEMS and ZVPT TACL definitions
-- must be loaded to compile this filter.
------------------------------------------------------------

[#def zems^val^ssid text |body|
      [zspi^val^TANDEM].[zspi^ssn^zems].0]
[#def zvpt^val^ssid text |body|
      [zspi^val^TANDEM].[zspi^ssn^zvpt].0]


[#DEF init^template^key STRUCT
  BEGIN
  STRUCT fields;
    BEGIN
    SSID template^ssid  value [zvpt^val^ssid];
    INT2 template^code  value [zvpt^tkn^inittemplate];
    INT2 template^value value [zvpt^val^inittemplate^null];
    END;
  BYTE  numbers(0:19) REDEFINES fields;
  END;
 ]

[#def InsertCommas macro |body|
 [#delta/commands ed^InsertCommas/%*%]
]

[#def ed^InsertCommas delta |body|
J<:S $;-DI,$>
]
```

Figure E-2. D-Series Default Alternate-Events Filter (page 2 of 4)

```
------------------------------------------------------------
-- We need to #def our own replica of an ssid structure.
-- This is necessary because the one in ZSPITACL does not
-- have the fields broken out; instead that structure just
-- consists of a single field of data type ssid.  We need
-- to refer to the fields separately in this filter.
------------------------------------------------------------

 [#def zspi_ddl_ssid STRUCT
    BEGIN
    CHAR Z^OWNER(0:7);
    INT  Z^NUMBER;
    UINT Z^VERSION;
    END;
 ]

FILTER Viewpt^alt^default^filter (
  ssid(zvpt^val^ssid, zvpt^fltr^select       )  OPTIONAL,
  ssid(zvpt^val^ssid, zvpt^fltr^discard      )  OPTIONAL,
  ssid(zvpt^val^ssid, zvpt^fltr^systemname   )  OPTIONAL,
  ssid(zvpt^val^ssid, zvpt^fltr^ssid         )  OPTIONAL,
  ssid(zvpt^val^ssid, zvpt^fltr^eventnumber  )  OPTIONAL,
  ssid(zvpt^val^ssid, zvpt^fltr^cpu          )  OPTIONAL,
  ssid(zvpt^val^ssid, zvpt^fltr^pin          )  OPTIONAL,
  ssid(zvpt^val^ssid, zvpt^fltr^processname  )  OPTIONAL,
  ssid(zvpt^val^ssid, zvpt^fltr^eventtext    )  OPTIONAL,
  ssid(zvpt^val^ssid, zvpt^fltr^custom^number)  OPTIONAL,
  ssid(zvpt^val^ssid, zvpt^fltr^custom^file  )  OPTIONAL,
  ssid(zvpt^val^ssid, zvpt^fltr^custom^string)  OPTIONAL );

begin ssid(zvpt^val^ssid)
if  zvpt^fltr^select = [zspi^val^true] or
    zvpt^fltr^discard = [zspi^val^true]  then
   begin

   -- Test for matching system name (if provided).
   if ( not tokenpresent(zvpt^fltr^systemname) or
        zvpt^fltr^systemname =
        decompose(ssid(zems^val^ssid,
                       zems^tkn^xsenderid^pd),system name)
      ) and

   -- Test for matching subsystem id (if provided).
      ( not tokenpresent(zvpt^fltr^ssid) or
        (  zvpt^fltr^ssid.zspi_ddl_ssid:z^owner =
             ssid(zems^val^ssid,zspi^tkn^ssid).
             zspi_ddl_ssid:z^owner
           and zvpt^fltr^ssid.zspi_ddl_ssid:z^number =
             ssid(zems^val^ssid,zspi^tkn^ssid).
             zspi_ddl_ssid:z^number
        )
      ) and
```

Figure E-2. D-Series Default Alternate-Events Filter (page 3 of 4)

```
   -- Test for matching event number (if provided).
      ( not tokenpresent(zvpt^fltr^eventnumber) or
        zvpt^fltr^eventnumber =
        ssid(zems^val^ssid,zems^tkn^eventnumber)
      ) and

   -- Test for matching cpu (if provided).
      ( not tokenpresent(zvpt^fltr^cpu) or
        zvpt^fltr^cpu =
        ssid(zems^val^ssid,
             zems^tkn^xsenderid).zems^ddl^xsenderid:zcpu
      ) and

   -- Test for matching pin (if provided).
      ( not tokenpresent(zvpt^fltr^pin) or
        zvpt^fltr^pin =
        ssid(zems^val^ssid,
             zems^tkn^xsenderid).zems^ddl^xsenderid:zpin
      ) and

   -- Test for matching process name (if provided).
      ( not tokenpresent(zvpt^fltr^processname) or
        decompose(zvpt^fltr^processname,
                  destination name, name part) =
        decompose(ssid(zems^val^ssid,
                  zems^tkn^xsenderid^pd),
                  destination name,name part)
      ) and

   -- Test for matching event text (if provided).
      ( not tokenpresent(zvpt^fltr^eventtext) or
        emstextmatch((
         [insertcommas [init^template^key:numbers(0:19)]]),
                     zvpt^fltr^eventtext)
      )

      --
      --  and ...
      --  user may add custom conditions here.
      --
   then
      -- conditions specified were met.
      begin
      if  zvpt^fltr^discard = [zspi^val^true] then fail;
      end

 else
      -- conditions specified were not met
      begin
      if  zvpt^fltr^select = [zspi^val^true] then fail;
      end;

   end; -- select or discard conditions
```

Figure E-2. D-Series Default Alternate-Events Filter (page 4 of 4)

```
 ------------------------------------------------------------
-- Events which have passed the conditions above will be
-- processed using the ViewPoint default filtering rules:
-- 1) Events with a suppress^display token value equal to
--    true fail unless the event is an action
--    completion event -- then it will pass with a pass
--    code value of 3.
-- 2) Events with an action^needed token are passed
--    with a pass code of 1.
-- 3) Events with an emphasis token value which is not
--    false are passed with a pass code 2.
-- 4) All other events pass with no pass code.
------------------------------------------------------------

begin ssid(zems^val^ssid)   -- zems ssid section.
-- fail on suppress^display events which are not
-- action-completion.

if zems^tkn^suppress^display = [zspi^val^true] then
   begin
   if tokenpresent (zems^tkn^action^needed) and
      zems^tkn^action^needed = [zspi^val^false] then pass 3
   else
      fail;
   end;

-- events which have not been discarded by above conditions
-- are returned to ViewPoint.  Action and critical events
-- are identified by the following conditions:

-- action events are identified by the action^needed token.
if tokenpresent (zems^tkn^action^needed) then pass 1;
-- critical events are identified by the emphasis token.
if zems^tkn^emphasis <> [zspi^val^false] then pass 2;

end;   -- zems ssid section.
pass;
end;
```

Glossary

Introduction

This glossary defines technical terms used in this manual. Some of these terms are also defined in the *SPI Programming Manual*.

abend (abnormal end). (1) An abnormal end of a task or process. (2) An error condition that can result in the termination of a program.

action event. An event for which the generating subsystem requires a response from the operator, such as mounting a tape. Each subsystem determines what events are action events by including a unique EMS token in the event message.

advice line. Line 21 (or 20 in some cases) of the ViewPoint block-mode screens. It provides information pertinent to your operation of the ViewPoint application, such as error notices and advisory messages (such as operation-complete messages). See Appendix B for a complete listing of such messages.

alternate event. An event selected for display to a particular operator according to criteria specified by that operator. Alternate events are displayed on the ViewPoint Alternate Events screen. Contrast with *Primary event*.

alternate-events distributor. An EMS consumer distributor that distributes event messages to a ViewPoint event collection server for display only to a particular operator. See also *Distributor*.

Alternate Event Configuration screen. The two-page ViewPoint Event Configuration screen requested from the Alternate Events screen that allows operators to control events sent to their Alternate Events screens.

Alternate Events screen. The screen on which ViewPoint displays events configured specifically for individual operators; these are events that pass the filter for alternate events and the configuration criteria specified on the Alternate Events Configuration screen. If not specifically configured, this screen shows the same events as the Primary Events screen.

background process. A server process that runs in the background—that is, a process that continues running until it is explicitly stopped. Background processes are defined and executed using the TACL routines that constitute the Define Process library. See also *Define Process library*.

buffer. Temporary storage of data. For instance, data to be sent in an interprocess message is encoded in a buffer from which it is copied by the file system. The data is delivered to a buffer addressable by the recipient. See also *SPI buffer*.

built-in. A primitive function or variable in TACL. Names of built-ins always begin with a pound sign (#).

clipboard file. An EDIT file to which you can copy selected lines from a ViewPoint display screen. You, or a management application program, can then access the selected lines of text in that file at some later time. Copying to the clipboard file is called *clipping*.

command. A demand for action by or information from a subsystem, or the operation demanded by an operator or application. A command is typically conveyed as an interprocess message from an application to a subsystem.

configuration file. A file that contains only configuration data. The ViewPoint application has configuration files for status and for events, and there may be several of each. An operator selects the desired configuration file by entering its name in the appropriate field in the Profile screen. When the operator exits from the Profile screen, the data in the named file is the new configuration for all status and event screens that are subsequently displayed.

control and inquiry. Those aspects of object management related to the state or configuration of an object. Such aspects include actions that affect the state or configuration of an object, inquiries about the object, and commands pertaining to the session environment (for example, commands that set default values for the session). Compare *Event management*.

critical event. An event designated as critical to the operation of the system or network. Each subsystem determines what set of events generated by that subsystem should be designated as critical. The default event-message filter delivered with the ViewPoint application classifies events as critical if they contain an EMS emphasis token (ZEMS-TKN-EMPHASIS) with a value of TRUE. This filter can be altered by the user.

Define Process library. A set of TACL routines that allows the user to run background server processes—that is, server processes that continue running until they are explicitly stopped—so that management applications can send commands to a number of subsystems without the overhead of creating a new server process for each command. The library includes routines to define run parameters for a background process, to start and stop the process, to manage its input and output, and to clear the input queue or the output queue for the process.

display type. The name of a status item that is available from a particular status server.

Distributed Systems Management (DSM). A set of software tools that facilitates management of Compaq NonStop™ Himilaya systems and Expand networks. These tools include the ViewPoint console application, the Subsystem Control Facility (SCF) for data-communications subsystems, the Subsystem Programmatic Interface (SPI), EMS, the Distributed Name Service (DNS), and token-oriented programmatic interfaces to the management processes for various Compaq *NonStop*™ Kernel subsystems.

distributor. An EMS process that distributes event messages from event log files to specific processes in the ViewPoint application. The application uses two kinds of distributors: forwarding event distributors, which forward event messages from remote nodes to a network control node, and consumer distributors, which distribute event messages to an event collection server. The consumer distributors are identified in the ViewPoint

application as the primary-events distributor, the alternate-events distributor, and the last-events distributor.

DSM. See *Distributed Systems Management*.

event. A significant change in some condition in the system or network. Events can be operational errors, notifications of limits exceeded, requests for action needed, and so on. Events are reported to ViewPoint as event messages to be displayed on a Primary Events, Alternate Events, or Last Events screen.

event cache. A file stored in memory that retains a set of event messages for quick access by the screen display modes. In the case of current-events mode, the events retained are the most recently received ones. In the case of historical-events mode, the events retained are those that follow the specified starting time.

event collection server. A ViewPoint server process that receives event messages from a primary-events or alternate-events distributor for display on a ViewPoint Primary Events, Alternate Events, or Last Events screen.

event collector. An EMS process that collects all operator event messages generated in a given system and files them in a file called an event log.

event configuration file. A file that contains configuration information to control what ViewPoint displays on an events screen. There can be any number of event configuration files; operators specify on the Profile screen which file ViewPoint is to use to display events at their terminals.

Event Configuration screen. Either the Primary Event Configuration or the Alternate Event Configuration screen; which screen ViewPoint displays is determined by whether event configuration was requested from the Primary Events or the Alternate Events screen. See also *Primary Event Configuration screen* and *Alternate Event Configuration screen*.

Event Detail screen. The ViewPoint screen on which selected event messages are described in detail.

event management. The reporting and logging of important events that occur in a system or network, the distribution and retrieval of information concerning those events, and the actions taken by operations personnel or software in response to the events. Compare *Control and inquiry*.

event message. The text intended for a system operator that describes a change in some condition in the system or network, whether minor or serious. The change of condition is termed an event. Events may be operational errors, notifications of limits exceeded, requests for action needed, and so on. Each event message, when displayed on an event screen, is displayed as one line of text.

Extras. A SCREEN COBOL program unit that a programmer can add to the ViewPoint application, possibly as a means of invoking a larger set of program units that you

supply. The ViewPoint application calls the Extras program unit (which must have the name EXTRAS) when the user presses the Extras function key.

filter. A file that contains a list of criteria against which incoming event messages can be compared so as to pass a given message (if it meets all criteria) or not pass it (if it fails one or more criteria).

filter object file. A file containing the compiled filter code that determines which events are displayed on an events screen.

high PIN (process identification number). A PIN that is greater than 255. See also *low PIN*.

historical mode. A view of event messages that has a specified starting time (or date and time) and a finite ending time, usually determined by the size of an event cache. If no starting time is specified, the view is one of current events rather than historical events.

last event. A recent event that concerns a particular object. When requested, ViewPoint displays on the Last Events screen a page of last-event messages whose subject is a selected object.

Last Events screen. The screen on which ViewPoint displays all events with a particular subject.

low PIN (process identification number). A PIN that is ranges from 0 to 254. See also *high PIN*.

macro. A sequence of TACL commands and built-in functions that can contain dummy arguments, thus providing a means for simple argument substitution. No validity checking of the arguments is performed. When the macro name is given to TACL, TACL substitutes the expansion of the command sequence for the name, replacing any dummy arguments with parameter values supplied in the invocation.

management application. A program or set of programs that issues commands to subsystems and/or retrieves event messages to assist in managing a computer system or a network of systems. A management application is a requester with respect to the subsystems to which it sends commands; the subsystems are servers with respect to the management application.

management process. The process through which an application issues commands to a subsystem. A management process can be associated with more than one subsystem; in this case, the management process is logically part of each of the subsystems. The Subsystem Control Point (SCP) and PATHMON are examples of management processes.

message. A block of information, usually in the form of a structure, that is sent from one process to another. See also *SPI message*.

Network Control Node (NCN). In the context of the ViewPoint application, a node (one Compaq system) where the ViewPoint terminal control process (TCP) is running. Since

it is possible to have more than one such process in a network, there can likewise be more than one NCN in a network of Compaq systems.

Network Status Summary screen. The screen on which ViewPoint displays information about the status of objects in a network.

object. In SPI, an entity subject to independent reference and control by a subsystem: for example, the disk file $DATA or the data-communications line $X2502. An object typically has a name and a type known to the controlling subsystem. There might also be aliases for the name and type in the DNS database. In DDL, an item in a dictionary. DDL assigns each object a unique object number for identification.

option line. Line 24 of the block-mode screens in the ViewPoint application. This line is used by operators to enter information (an option) that is to be used by the next function key that will be pressed.

outstanding event. An action or critical event that has not yet been deleted by operator action or by system action. The outstanding attribute is created by the ViewPoint application for management of important events; it is not an inherent part of the event message.

passthrough mode. In Define Process, the direct interaction with a defined process, entering commands to its prompts, as opposed to allowing TACL to pass the commands to the defined process.

primary event. An event that is reported to a collector for display at all ViewPoint terminals at a network control node. Primary events are displayed on the ViewPoint Primary Events screen. See alternate event.

Primary-events distributor. An EMS consumer distributor that distributes event messages to a ViewPoint event collection server for display on all terminals at a network control node. See also *distributor*.

Primary Event Configuration screen. The one-page ViewPoint Event Configuration screen requested from the Primary Events screen on which the operator can specify criteria to control the display of events on the Primary Events screen for all operators.

Primary Events screen. The screen on which ViewPoint displays events configured for all operators; these are events that pass the filter for primary events and the configuration criteria specified on the Primary Events Configuration screen.

profile screen. The ViewPoint screen on which the operator can specify the status configuration and event configuration files to be used to control the display of status on the Network Status Summary screen and of events on the Primary Events screen.

programmatic command. A command issued by an application program rather than by a human operator.

programmatic interface. A means for a program to communicate with another program. On a Compaq system, a programmatic interface typically includes the following: a message

format, a set of message formats, or a set of procedures (such as the SPI procedures) to build and decode messages; definitions of message elements (commands, data types, objects, parameters, response data, errors, and so on); rules for communication between the requester and the server; and software to receive and respond to messages defined for the interface.

programmed operator. A management application that performs functions that might otherwise be performed by a human operator.

reply, reply message. One message from a server to a requester, in reaction to a request.

request, request message. One message from a requester to a server.

response. The information or confirmation supplied by a subsystem in reaction to a command. A response is typically conveyed as one or more interprocess messages (reply messages) from a subsystem to an application.

server classes. One or more processes sharing the work load but having a single server class name.

session. The period during which two entities can exchange data. In the context of a management application, the period during which an application can issue commands to a subsystem. In the context of a command interpreter, the period during which a user can issue commands to the command interpreter. In the context of the ViewPoint application, the period between the user's invoking ViewPoint and exiting.

SPI. See *Subsystem Programmatic Interface*.

SPI buffer. A sequence of memory locations containing a message produced by the SPI procedures.

SPI message. A message specially formatted by the SPI procedures for communication between a management application and a subsystem, or between one subsystem and another. An SPI message consists of a collection of tokens. To retrieve a token from the message, the application passes a token code to SPI, which scans for the appropriate token and returns its value to the application. Note that an SPI message is a single block of information sent at one time as a unit (sent with one file-system procedure call if the file system is used). The complete transmittal of a command and the receipt of the response to it can consist of several exchanges of command and response messages.

SPI procedures. The set of operating system procedures used to build and decode buffers for use in system and network management and in certain other applications. These procedures are SSINIT, SSNULL, SSPUT, SSPUTTKN, SSGET, SSGETTKN, SSMOVE, and SSMOVETKN.

status collection server. See *status server*.

status configuration file. A file that contains the configuration information controlling what ViewPoint displays on the Network Status Summary screen. There can be any number

of status configuration files; operators specify on the Profile screen which file ViewPoint uses to display status at their terminals.

Status Configuration screen. The ViewPoint screen on which the operator can specify configuration information to control the display of status items on the Network Status Summary screen.

status item. A measurable condition in a system or network that is displayed on one line of a Network Status Summary display screen. Each such item is separately chosen and configured for display by use of the Status Item Configuration screen.

Status Item Configuration screen. The ViewPoint screen on which individual operators can modify existing items or add new items to a status configuration file for display on their Network Status Summary screen.

status server. A server that collects data for the ViewPoint Network Status Summary screen. Compaq provides a default status server; the user can write and configure additional status servers to augment or replace the functions of the default status server. Typically, a status server issues programmatic commands to subsystems to find out how many or how much of some resource is available—for example, how many network nodes are available or how many terminals are up. A status server is defined to Pathway as a server class. A status server can also be called a status collection server.

subject. In event management, a device, process, or other named entity about which a given event message is concerned.

subsystem. A program or set of processes that manages a cohesive set of objects. Each subsystem has a management process through which applications can request services by issuing commands defined by that subsystem. See management process.

subsystem ID. A data structure that uniquely identifies a subsystem (including whether it is a Tandem subsystem or a user subsystem). It consists of the name of the owner of the subsystem (the company that provides it), a subsystem number that denotes the subsystem within the scope of its owner, and a subsystem version number. The subsystem ID is an argument to most of the SPI procedures.

subsystem number. An integer that identifies a subsystem within the context of its owner. The subsystem owner, the subsystem number, and the subsystem version number make up the subsystem ID that uniquely identifies a subsystem.

Subsystem Programmatic Interface. A common, message-based interface that can be used to build and decode messages used for communication between requesters and servers—for instance, in a management application. It includes procedures to build and decode specially formatted messages (as described under “SPI message”); definition files in TAL, COBOL, and TACL for inclusion in programs, macros, and routines using the interface procedures; and definition files in DDL for programmers writing their own subsystems.

token. In SPI, a distinguishable unit in an SPI message. Programs place tokens in an SPI buffer using the SSPUT procedure (except for header tokens, which are a special case)

and retrieve them from the buffer with the SSGET procedure. A token has two parts: an identifying code—a token code or token map—and a value. For control and inquiry, a token normally represents a parameter to a command, an item of information in a response, or control information for SPI or the subsystem. For event management, a token normally represents an item of information about an event, or control information for EMS or the subsystem.

token code. In SPI, a 32-bit number, or the variable name representing that number, that consists of a token type and a token number. Most token codes are used to identify tokens, but some are used instead to designate special operations by the SPI procedures, such as obtaining token attributes (for instance, count, address, and token length) or changing position or context within the SPI buffer. When used to identify a token, a token code defines a simple data type or structure that the developer of the subsystem has declared cannot change in future releases.

token value. The value assigned to a token.

Index

A

B

C

C (continued)

D

D (continued)

E

E (continued)

E (continued)

F

F (continued)

G

H

I

I (continued)

J

L

M

M (continued)

N

O

P

P

P (continued)

Q

R

R (continued)

S

S (continued)

S (continued)

T

T (continued)

U

V

V (continued)

W

X

Z

Z (continued)

Special Characters